Network Working Group
Request for Comments: 1321

R. Rivest
MIT Laboratory for Computer Science
and RSA Data Security, Inc.

April 1992

Алгоритм MD5

The MD5 Message-Digest Algorithm

Статус документа

Этот документ содержит информацию для сообщества Internet. Документ не задаёт каких-либо стандартов. Разрешается свободное распространение документа.

Благодарности

Автор благодарит Don Coppersmith, Burt Kaliski, Ralph Merkle, David Chaum и Noam Nisan за множество полезных комментариев и предложений.

Оглавление

1. Введение

В этом документе описывается алгоритм цифровых подписей[1] MD5. Этот алгоритм принимает на входе сообщение произвольного размера и выдает в результате 128-битовый «отпечаток» (fingerprint) или цифровую подпись (message digest). Предполагается, что потребуются нереальный объем вычислений для создания двух сообщений, цифровые подписи которых совпадут, или подбора сообщения по имеющейся цифровой подписи. Алгоритм MD5 предназначен для создания цифровой подписи (сигнатуры), когда требуется безопасно «сжать» большой файл перед тем, как шифровать его с использованием закрытого (секретного) ключа в системах с открытыми ключами типа RSA.

Алгоритм MD5 разработан для обеспечения достаточной скорости на 32-битовых машинах. В дополнение к этому MD5 не требует использования больших таблиц подстановки; кодирование алгоритма может быть достаточно компактным.

MD5 является расширением алгоритма цифровых подписей MD4 [1,2]. MD5 несколько медленней, чем MD4, но устроен более «консервативно». Причиной разработки MD5 послужило ощущение того, что алгоритм MD4 может быть адаптирован для применения быстрее, нежели предполагалось; поскольку при разработке MD4 основным требованием была высокая скорость, он находится «на самом краю» допустимости в плане устойчивости к криптоаналитическим атакам. MD5 немного проигрывает в скорости, но существенно выигрывает в безопасности. Алгоритм MD5 является открытым и возможно будет адаптирован в качестве стандартного.

Для приложений OSI идентификатор объекта MD5 имеет форму

```
md5 OBJECT IDENTIFIER ::= iso(1) member-body(2) US(840) rsadsi(113549) digestAlgorithm(2) 5}
```

В идентификаторе типа X.509 (AlgorithmIdentifier [3]) параметры для MD5 должны иметь тип NULL.

2. Термины и обозначения

В этом документе термин «слово» (word) используется для 32-битовых значений, а «байт» - для восьмибитовых. Последовательности битов могут интерпретироваться естественным путем как последовательность байтов, в которой каждая группа из 8 последовательных битов интерпретируется как байт, гле старший (наиболее значимый) бит располагается первым. Подобно этому последовательность байтов может интерпретироваться как последовательность 32-битовых слов, в которой каждая группа из 4 последовательных байтов интерпретируется как слово, где младший (наименее значимый) указывается первым.

[1]Message-digest algorithm.

Запись **x_i** означает x i-ое (x[i]). Если индекс представляет собой выражение, оно будет заключаться в фигурные скобки **x_{i+1}**. Символ ^ используется для обозначения степени - x^i означает x в степени i.

Символ «+» обозначает сложение слов (т. е., сложение с использованием модуля 2^{32}). Запись **X <<< s** означает 32-битовое слово, полученное циклическим сдвигом X влево на s позиций. **not(X)** означает побитовое дополнение X, а **X v Y** - побитовую операцию **X OR Y** (X ИЛИ Y). Запись **X xor Y** означает побитовую операцию **X XOR Y** (исключающее ИЛИ), а **XY** побитовую операцию **X AND Y** (X И Y).

3. Описание алгоритма MD5

Начнем с допущения о наличии на входе сообщения размером b битов, для которого нужно создать цифровую подпись. Здесь b означает неотрицательное целое число, b может принимать нулевое значение и не обязано быть кратным 8. Это значение может быть неограниченно большим. Биты исходного сообщения будем представлять следующим образом:

```
m_0 m_1 ... m_{b-1}
```

Для создания цифровой подписи выполняется процесс из 5 описанных ниже этапов.

3.1 Этап 1. Добавление битов заполнения

Сообщение дополняется до размера (в битах), конгруэнтного 448 по модулю 512. Т. е., размер сообщения устанавливается таким, чтобы после добавления 64 битов размер был кратен 512. Заполнение производится во всех случаях, даже если размер исходного сообщения конгруэнтен 448 по модулю 512.

Заполнение происходит следующим образом - сначала в конце сообщения добавляется один бит, имеющий значение 1, а потом добавляются биты, имеющие значение 0 до размера, конгруэнтного 448 по модулю 512. В любом случае число добавляемых битов не может быть меньше 1 и больше 512.

3.2 Этап 2. Добавление размера сообщения

64-битовое представление b (размер исходного сообщения в битах) добавляется в конец результата предыдущего этапа. В редких случаях, когда b больше 2^{64}, используются только младшие 64 бита b. Биты добавляются в конце, как два 32-битовых слова, в которых младший байт размещается в начале, как было условлено выше.

После этого размер полученного сообщения (исходное сообщение + заполнение + размер) будет кратен 512 битам. Следовательно, результирующее сообщение будет содержать целое число блоков по 16 слов (32 бита в каждом слове). Пусть M[0 ... N-1] обозначает слова полученного в результате сообщения (N кратно 16).

3.3 Этап 3. Инициализация буфера MD

Для расчета цифровой подписи используется буфер на 4 слова (A,B,C,D). Каждое из слов A, B, C, D представляет собой 32-битовый регистр. Эти регистры инициализируются приведенными ниже значениями (шестнадцатиричное представление, сначала младший байт):

```
слово A: 01 23 45 67
слово B: 89 ab cd ef
слово C: fe dc ba 98
слово D: 76 54 32 10
```

3.4 Этап 4. Обработка сообщения блоками по 16 слов

Сначала определим четыре дополнительных функции, каждая из которых принимает три 32-битовых слова в качестве аргументов и возвращает одно 32-битовое слово.

```
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))
```

Для каждого бита функция F работает как условие - если X, то Y, иначе Z. Функцию F можно определить с использованием сложения (+) вместо операции ИЛИ (v), поскольку XY и not(X)Z никогда не будут давать 1 в одной битовой позиции. Интересно отметить, что при условии, когда биты X, Y и Z независимы и не образованы смещением (unbiased), каждый бит функции F(X,Y,Z) будет независимым и не будет образован смещением.

Функции G, H, I похожи на функцию F - фактически они действуют «параллельно по битам» для создания результатов по значениям X, Y, Z так, чтобы при независимых и не образованных смещением битах X, Y и Z каждый бит G(X,Y,Z), H(X,Y,Z) и I(X,Y,Z) также был независимым и не получался путем смещения. Отметим, что функция H представляет собой побитовую операцию «исключающее ИЛИ» (xor) или функцию «четности» (parity) переданных ей аргументов.

На этом этапе используется 64-элементная таблица T[1 ... 64], построенная с использованием синусоидальной функции. Пусть T[i] обозначает i-й элемент таблицы, который равен целой части произведения 4294967296 * abs(sin(i)), где i задано в радианах. Элементы таблицы приведены в приложении.

Ниже показана обработка блоков сообщения.

```
/* обработать каждый блок из 16 слов. */
For i = 0 to N/16-1 do

        /* скопировать блок i в X. */
        For j = 0 to 15 do
                Set X[j] to M[i*16+j]
        end /* цикл по j */

        /* сохранить A как AA, B как BB, C как CC, D как DD. */
        AA = A
        BB = B
        CC = C
```

```
        DD = D

        /* Раунд 1. */
        /* [abcd k s i] обозначает операцию a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
        /* Выполним следующие 16 операций. */
        [ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
        [ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
        [ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
        [ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

        /* Раунд 2. */
        /* [abcd k s i] обозначает операцию a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
        /* Выполним следующие 16 операций. */
        [ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
        [ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
        [ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
        [ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]

        /* Раунд 3. */
        /* [abcd k s i[1]] обозначает операцию a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
        /* Выполним следующие 16 операций. */
        [ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
        [ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
        [ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
        [ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]

        /* Раунд 4. */
        /* [abcd k s i[1]] обозначает операцию a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
        /* Выполним следующие 16 операций. */
        [ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
        [ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
        [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
        [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

        /* После этого добавим к каждому из 4 регистров значение, которое этот регистр */
        /* имел до начала выполнения этого блока. */
        A = A + AA
        B = B + BB
        C = C + CC
        D = D + DD
end /* цикл по i */
```

3.5 Этап 5. Вывод

Сигнатура сообщения выводится как A, B, C, D (т. е., мы начинаем вывод с младшего байта A и закачиваем старшим байтом D).

На этом описание алгоритма MD5 завершается. Пример реализации на языке C дан в приложении.

4. Заключение

Алгоритм цифровых подписей MD5 прост в реализации и создает «отпечатки» (fingerprint) или цифровые подписи для сообщений произвольной длины. Предполагается, что для создания двух сообщений с одинаковыми сигнатурами потребуется порядка 2^{64} операций, а для подбора сообщения по имеющейся сигнатуре – порядка 2^{128} операций. Алгоритм MD5 был тщательно исследован на предмет поиска слабых мест. Однако алгоритм является достаточно новым и требуется дополнительный анализ, как и для всех новинок этого типа.

5. Различия между MD4 и MD5

Ниже перечисляются отличия алгоритмами MD5 от MD4.

1. Добавлен Раунд 4.
2. На каждом этапе используется уникальная аддитивная константа.
3. Функция G в раунде 2 была изменена с (XY v XZ v YZ) на (XZ v Y not(Z)), чтобы уменьшить симметрию G.
4. На каждом этапе добавляется результат предыдущего этапа. Это обеспечивает более быстрое возникновение «лавинного эффекта».
5. Изменен порядок доступа к входным словам для раундов 2 и 3, чтобы сделать результаты менее похожими.
6. Оптимизировано количество сдвигов в каждом раунде для достижения более быстрого «лавинного эффекта». Смещения в каждом раунде отличаются.

Литература

[1] Rivest, R., "The MD4 Message Digest Algorithm", RFC 1320, MIT and RSA Data Security, Inc., April 1992.

[2] Rivest, R., "The MD4 message digest algorithm", in A.J. Menezes and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90 Proceedings, pages 303-311, Springer-Verlag, 1991.

[3] CCITT Recommendation X.509 (1988), "The Directory - Authentication Framework."

[1] В оригинале допущена опечатка - вместо i указано t. *Прим. перев.*

Приложение A – Пример реализации

В этом приложении содержатся файлы, заимствованные из RSAREF: A Cryptographic Toolkit for Privacy-Enhanced Mail.

```
global.h – общий файл заголовков
md5.h - файл заголовков для MD5
md5c.c – исходный код MD5
```

Для получения дополнительной информации об RSAREF пишите по адресу <rsaref@rsa.com>. В приложение включен также файл

```
mddriver.c – драйвер для тестирования MD2, MD4 и MD5
```

Драйвер по умолчанию компилируется для проверки MD5 и может быть скомпилирован для проверки MD2 или MD4, если в командной строке компилятора C задать для MD значение 2 или 4, соответственно.

Реализация переносима и должна работать на разных платформах. Однако ее несложно оптимизировать для конкретной платформы (оставим это в качестве упражнения для читателей). Например, для платформ типа «little-endian», где байт с наименьшим адресом в 32-битовом слове является наименее значимым и нет ограничений по выравниванию, вызов Decode в MD5Transform можно заменить преобразованием типа.

A.1 global.h

```
/* GLOBAL.H – типы и константы RSAREF */

/* Для PROTOTYPES следует использовать значение 1 тогда и только тогда, когда компилятор
поддерживает прототипы аргументов в функциях. Приведенный ниже код устанавливает для PROTOTYPES по
умолчанию значение 0, если этот параметр уже не задан флагами компилятора C.
*/
#ifndef PROTOTYPES
#define PROTOTYPES 0
#endif

/* POINTER определяет базовый тип указателя */
typedef unsigned char *POINTER;

/* UINT2 определяет 2-байтовое слово */
typedef unsigned short int UINT2;

/* UINT4 определяет 4-байтовое слово */
typedef unsigned long int UINT4;

/* PROTO_LIST определяется в зависимости от определенного ранее значения PROTOTYPES.
При использовании PROTOTYPES список PROTO_LIST будет возвращать прототипы, иначе будет пустым.
*/
#if PROTOTYPES
#define PROTO_LIST(list) list
#else
#define PROTO_LIST(list) ()
#endif
```

A.2 md5.h

```
/* MD5.H – заголовочный файл для MD5C.C */

/* Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991.
All rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
*/

/* Контекст MD5. */
typedef struct {
        UINT4 state[4];              /* состояние (ABCD) */
        UINT4 count[2];              /* число битов, modulo 2^64 (сначала младший) */
        unsigned char buffer[64];    /* входной буфер*/
} MD5_CTX;

void MD5Init PROTO_LIST ((MD5_CTX *));
void MD5Update PROTO_LIST ((MD5_CTX *, unsigned char *, unsigned int));
void MD5Final PROTO_LIST ((unsigned char [16], MD5_CTX *));
```

A.3 md5c.c

```
/* MD5C.C - RSA Data Security, Inc., алгоритм цифровой подписи MD5 */

/* Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
*/

#include "global.h"
#include "md5.h"

/* Константы для подпрограммы MD5Transform. */

#define S11 7
#define S12 12
#define S13 17
#define S14 22
#define S21 5
#define S22 9
#define S23 14
#define S24 20
#define S31 4
#define S32 11
#define S33 16
#define S34 23
#define S41 6
#define S42 10
#define S43 15
#define S44 21

static void MD5Transform PROTO_LIST ((UINT4 [4], unsigned char [64]));
static void Encode PROTO_LIST ((unsigned char *, UINT4 *, unsigned int));
static void Decode PROTO_LIST ((UINT4 *, unsigned char *, unsigned int));
static void MD5_memcpy PROTO_LIST ((POINTER, POINTER, unsigned int));
static void MD5_memset PROTO_LIST ((POINTER, int, unsigned int));

static unsigned char PADDING[64] = {
  0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};

/* F, G, H и I являются базовыми функциями MD5. */
#define F(x, y, z) (((x) & (y)) | ((~x) & (z)))
#define G(x, y, z) (((x) & (z)) | ((y) & (~z)))
#define H(x, y, z) ((x) ^ (y) ^ (z))
#define I(x, y, z) ((y) ^ ((x) | (~z)))

/* ROTATE_LEFT циклически смещает x влево на n битов. */
#define ROTATE_LEFT(x, n) (((x) << (n)) | ((x) >> (32-(n))))

/* Преобразования FF, GG, HH и II для раундов 1, 2, 3, 4.
Циклический сдвиг отделен от добавления для предотвращения повторного расчета.
*/
#define FF(a, b, c, d, x, s, ac) { \
        (a) += F ((b), (c), (d)) + (x) + (UINT4)(ac); \
        (a) = ROTATE_LEFT ((a), (s)); \
        (a) += (b); \
}
#define GG(a, b, c, d, x, s, ac) { \
        (a) += G ((b), (c), (d)) + (x) + (UINT4)(ac); \
        (a) = ROTATE_LEFT ((a), (s)); \
        (a) += (b); \
}
#define HH(a, b, c, d, x, s, ac) { \
        (a) += H ((b), (c), (d)) + (x) + (UINT4)(ac); \
        (a) = ROTATE_LEFT ((a), (s)); \
        (a) += (b); \
```

```
}
#define II(a, b, c, d, x, s, ac) { \
        (a) += I ((b), (c), (d)) + (x) + (UINT4)(ac); \
        (a) = ROTATE_LEFT ((a), (s)); \
        (a) += (b); \
}

/* Инициализация MD5. Начинает работу MD5, записывая новый контекст. */
void MD5Init (context)
MD5_CTX *context;                                        /* контекст */
{
        context->count[0] = context->count[1] = 0;
        /* загрузка "магических" констант инициализации. */
        context->state[0] = 0x67452301;
        context->state[1] = 0xefcdab89;
        context->state[2] = 0x98badcfe;
        context->state[3] = 0x10325476;
}

/* MD5 блокирует операцию обновления. Продолжается создание сигнатуры MD5, обработка другого
  блока сообщения и обновление контекста.
*/
void MD5Update (context, input, inputLen)
MD5_CTX *context;                               /* контекст */
unsigned char *input;                           /* входной блок */
unsigned int inputLen;                          /* размер входного блока */
{
        unsigned int i, index, partLen;

        /* Расчет числа байтов mod 64 */
        index = (unsigned int)((context->count[0] >> 3) & 0x3F);

        /* Обновление числа битов */
        if ((context->count[0] += ((UINT4)inputLen << 3)) < ((UINT4)inputLen << 3))
                context->count[1]++;
        context->count[1] += ((UINT4)inputLen >> 29);

        partLen = 64 - index;

        /* Преобразование возможное число раз. */
        if (inputLen >= partLen) {
                MD5_memcpy((POINTER)&context->buffer[index], (POINTER)input, partLen);
                MD5Transform (context->state, context->buffer);

                for (i = partLen; i + 63 < inputLen; i += 64)
                        MD5Transform (context->state, &input[i]);

                index = 0;
        } else
                i = 0;
        /* Буферизация оставшихся входных данных */
        MD5_memcpy((POINTER)&context->buffer[index], (POINTER)&input[i], inputLen-i);
}

/* Завершение работы MD5. Окончание расчета цифровой подписи MD5, вывод сигнатуры и обнуление
   контекста.
*/
void MD5Final (digest, context)
unsigned char digest[16];                       /* цифровая подпись */
MD5_CTX *context;                               /* контекст */
{
        unsigned char bits[8];
        unsigned int index, padLen;

        /* Сохранение числа битов */
        Encode (bits, context->count, 8);

        /* Заполнение до 56 mod 64. */
        index = (unsigned int)((context->count[0] >> 3) & 0x3f);
        padLen = (index < 56) ? (56 - index) : (120 - index);
        MD5Update (context, PADDING, padLen);

        /* Добавление размера (до заполнения) */
        MD5Update (context, bits, 8);

        /* Сохранение состояния в дайджесте */
        Encode (digest, context->state, 16);

        /* Обнуление чувствительной информации. */
        MD5_memset ((POINTER)context, 0, sizeof (*context));
}

/* Базовое преобразование MD5. Преобразует состояние поблочно. */
static void MD5Transform (state, block)
UINT4 state[4];
unsigned char block[64];
```

```
{
        UINT4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

        Decode (x, block, 64);

        /* Раунд 1 */
        FF (a, b, c, d, x[ 0], S11, 0xd76aa478); /* 1 */
        FF (d, a, b, c, x[ 1], S12, 0xe8c7b756); /* 2 */
        FF (c, d, a, b, x[ 2], S13, 0x242070db); /* 3 */
        FF (b, c, d, a, x[ 3], S14, 0xc1bdceee); /* 4 */
        FF (a, b, c, d, x[ 4], S11, 0xf57c0faf); /* 5 */
        FF (d, a, b, c, x[ 5], S12, 0x4787c62a); /* 6 */
        FF (c, d, a, b, x[ 6], S13, 0xa8304613); /* 7 */
        FF (b, c, d, a, x[ 7], S14, 0xfd469501); /* 8 */
        FF (a, b, c, d, x[ 8], S11, 0x698098d8); /* 9 */
        FF (d, a, b, c, x[ 9], S12, 0x8b44f7af); /* 10 */
        FF (c, d, a, b, x[10], S13, 0xffff5bb1); /* 11 */
        FF (b, c, d, a, x[11], S14, 0x895cd7be); /* 12 */
        FF (a, b, c, d, x[12], S11, 0x6b901122); /* 13 */
        FF (d, a, b, c, x[13], S12, 0xfd987193); /* 14 */
        FF (c, d, a, b, x[14], S13, 0xa679438e); /* 15 */
        FF (b, c, d, a, x[15], S14, 0x49b40821); /* 16 */

        /* Раунд 2 */
        GG (a, b, c, d, x[ 1], S21, 0xf61e2562); /* 17 */
        GG (d, a, b, c, x[ 6], S22, 0xc040b340); /* 18 */
        GG (c, d, a, b, x[11], S23, 0x265e5a51); /* 19 */
        GG (b, c, d, a, x[ 0], S24, 0xe9b6c7aa); /* 20 */
        GG (a, b, c, d, x[ 5], S21, 0xd62f105d); /* 21 */
        GG (d, a, b, c, x[10], S22,  0x2441453); /* 22 */
        GG (c, d, a, b, x[15], S23, 0xd8a1e681); /* 23 */
        GG (b, c, d, a, x[ 4], S24, 0xe7d3fbc8); /* 24 */
        GG (a, b, c, d, x[ 9], S21, 0x21e1cde6); /* 25 */
        GG (d, a, b, c, x[14], S22, 0xc33707d6); /* 26 */
        GG (c, d, a, b, x[ 3], S23, 0xf4d50d87); /* 27 */
        GG (b, c, d, a, x[ 8], S24, 0x455a14ed); /* 28 */
        GG (a, b, c, d, x[13], S21, 0xa9e3e905); /* 29 */
        GG (d, a, b, c, x[ 2], S22, 0xfcefa3f8); /* 30 */
        GG (c, d, a, b, x[ 7], S23, 0x676f02d9); /* 31 */
        GG (b, c, d, a, x[12], S24, 0x8d2a4c8a); /* 32 */

        /* Раунд 3 */
        HH (a, b, c, d, x[ 5], S31, 0xfffa3942); /* 33 */
        HH (d, a, b, c, x[ 8], S32, 0x8771f681); /* 34 */
        HH (c, d, a, b, x[11], S33, 0x6d9d6122); /* 35 */
        HH (b, c, d, a, x[14], S34, 0xfde5380c); /* 36 */
        HH (a, b, c, d, x[ 1], S31, 0xa4beea44); /* 37 */
        HH (d, a, b, c, x[ 4], S32, 0x4bdecfa9); /* 38 */
        HH (c, d, a, b, x[ 7], S33, 0xf6bb4b60); /* 39 */
        HH (b, c, d, a, x[10], S34, 0xbebfbc70); /* 40 */
        HH (a, b, c, d, x[13], S31, 0x289b7ec6); /* 41 */
        HH (d, a, b, c, x[ 0], S32, 0xeaa127fa); /* 42 */
        HH (c, d, a, b, x[ 3], S33, 0xd4ef3085); /* 43 */
        HH (b, c, d, a, x[ 6], S34,  0x4881d05); /* 44 */
        HH (a, b, c, d, x[ 9], S31, 0xd9d4d039); /* 45 */
        HH (d, a, b, c, x[12], S32, 0xe6db99e5); /* 46 */
        HH (c, d, a, b, x[15], S33, 0x1fa27cf8); /* 47 */
        HH (b, c, d, a, x[ 2], S34, 0xc4ac5665); /* 48 */

        /* Раунд 4 */
        II (a, b, c, d, x[ 0], S41, 0xf4292244); /* 49 */
        II (d, a, b, c, x[ 7], S42, 0x432aff97); /* 50 */
        II (c, d, a, b, x[14], S43, 0xab9423a7); /* 51 */
        II (b, c, d, a, x[ 5], S44, 0xfc93a039); /* 52 */
        II (a, b, c, d, x[12], S41, 0x655b59c3); /* 53 */
        II (d, a, b, c, x[ 3], S42, 0x8f0ccc92); /* 54 */
        II (c, d, a, b, x[10], S43, 0xffeff47d); /* 55 */
        II (b, c, d, a, x[ 1], S44, 0x85845dd1); /* 56 */
        II (a, b, c, d, x[ 8], S41, 0x6fa87e4f); /* 57 */
        II (d, a, b, c, x[15], S42, 0xfe2ce6e0); /* 58 */
        II (c, d, a, b, x[ 6], S43, 0xa3014314); /* 59 */
        II (b, c, d, a, x[13], S44, 0x4e0811a1); /* 60 */
        II (a, b, c, d, x[ 4], S41, 0xf7537e82); /* 61 */
        II (d, a, b, c, x[11], S42, 0xbd3af235); /* 62 */
        II (c, d, a, b, x[ 2], S43, 0x2ad7d2bb); /* 63 */
        II (b, c, d, a, x[ 9], S44, 0xeb86d391); /* 64 */

        state[0] += a;
        state[1] += b;
        state[2] += c;
        state[3] += d;

        /* Обнуление чувствительных данных. */
        MD5_memset ((POINTER)x, 0, sizeof (x));
}
```

```
/* Кодирование входных данных (UINT4) в выходные (unsigned char).
   Предполагается, что размер кратен 4.
*/
static void Encode (output, input, len)
unsigned char *output;
UINT4 *input;
unsigned int len;
{
        unsigned int i, j;

        for (i = 0, j = 0; j < len; i++, j += 4) {
                output[j] = (unsigned char)(input[i] & 0xff);
                output[j+1] = (unsigned char)((input[i] >> 8) & 0xff);
                output[j+2] = (unsigned char)((input[i] >> 16) & 0xff);
                output[j+3] = (unsigned char)((input[i] >> 24) & 0xff);
        }
}

/* Декодирование входных данных (unsigned char) в выходные (UINT4).
   Предполагается, что размер кратен 4.
*/
static void Decode (output, input, len)
UINT4 *output;
unsigned char *input;
unsigned int len;
{
        unsigned int i, j;

        for (i = 0, j = 0; j < len; i++, j += 4)
                output[i] = ((UINT4)input[j]) | (((UINT4)input[j+1]) << 8) |
                            (((UINT4)input[j+2]) << 16) | (((UINT4)input[j+3]) << 24);
}

/* Примечание: По возможности замените цикл for стандартной функцией memcpy. */
static void MD5_memcpy (output, input, len)
POINTER output;
POINTER input;
unsigned int len;
{
        unsigned int i;

        for (i = 0; i < len; i++)
                output[i] = input[i];
}

/* Примечание: По возможности замените цикл for стандартной функцией memset. */
static void MD5_memset (output, value, len)
POINTER output;
int value;
unsigned int len;
{
        unsigned int i;

        for (i = 0; i < len; i++)
                ((char *)output)[i] = (char)value;
}
```

A.4 mddriver.c

```
/* MDDRIVER.C - тестовый драйвер для MD2, MD4 и MD5 */

/* Copyright (C) 1990-2, RSA Data Security, Inc. Created 1990.
All rights reserved.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
*/

/* Включенное ниже определение по умолчанию обеспечивает проверку MD5, если флаги компилятора
   С не задают иного.
*/
#ifndef MD
#define MD 5
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#include "global.h"
#if MD == 2
#include "md2.h"
```

```
#endif
#if MD == 4
#include "md4.h"
#endif
#if MD == 5
#include "md5.h"
#endif

/* Размер и число тестовых блоков. */
#define TEST_BLOCK_LEN 1000
#define TEST_BLOCK_COUNT 1000

static void MDString PROTO_LIST ((char *));
static void MDTimeTrial PROTO_LIST ((void));
static void MDTestSuite PROTO_LIST ((void));
static void MDFile PROTO_LIST ((char *));
static void MDFilter PROTO_LIST ((void));
static void MDPrint PROTO_LIST ((unsigned char [16]));

#if MD == 2
#define MD_CTX MD2_CTX
#define MDInit MD2Init
#define MDUpdate MD2Update
#define MDFinal MD2Final
#endif
#if MD == 4
#define MD_CTX MD4_CTX
#define MDInit MD4Init
#define MDUpdate MD4Update
#define MDFinal MD4Final
#endif
#if MD == 5
#define MD_CTX MD5_CTX
#define MDInit MD5Init
#define MDUpdate MD5Update
#define MDFinal MD5Final
#endif

/* Драйвер.
Аргументы (допускаются любые комбинации):
  -sstring - строка сигнатур
  -t       - определять время
  -x       - использовать тестовый сценарий
  filename - файл сигнатур
  (none)   - стандартный ввод для сигнатур
 */
int main (argc, argv)
int argc;
char *argv[];
{
        int i;

        if (argc > 1)
                for (i = 1; i < argc; i++)
                        if (argv[i][0] == '-' && argv[i][1] == 's')
                                MDString (argv[i] + 2);
                        else if (strcmp (argv[i], "-t") == 0)
                                MDTimeTrial ();
                        else if (strcmp (argv[i], "-x") == 0)
                                MDTestSuite ();
                        else
                                MDFile (argv[i]);
        else
                MDFilter ();

        return (0);
}

/* Создает и выводит сигнатуру. */
static void MDString (string)
char *string;
{
        MD_CTX context;
        unsigned char digest[16];
        unsigned int len = strlen (string);

        MDInit (&context);
        MDUpdate (&context, string, len);
        MDFinal (digest, &context);

        printf ("MD%d (\"%s\") = ", MD, string);
        MDPrint (digest);
        printf ("\n");
}

/* Измерение времени создания сигнатуры для TEST_BLOCK_COUNT блоков по TEST_BLOCK_LEN байтов. */
```

```
static void MDTimeTrial ()
{
        MD_CTX context;
        time_t endTime, startTime;
        unsigned char block[TEST_BLOCK_LEN], digest[16];
        unsigned int i;
        printf ("MD%d time trial. Digesting %d %d-byte blocks ...", MD,
                TEST_BLOCK_COUNT, TEST_BLOCK_LEN[1]);

        /* Инициализация блока */
        for (i = 0; i < TEST_BLOCK_LEN; i++)
                block[i] = (unsigned char)(i & 0xff);

        /* Запуск таймера */
        time (&startTime);

        /* Создание сигнатур блоков */
        MDInit (&context);
        for (i = 0; i < TEST_BLOCK_COUNT; i++)
                MDUpdate (&context, block, TEST_BLOCK_LEN);
        MDFinal (digest, &context);

        /* Остановка таймера */
        time (&endTime);

        printf (" done\n");
        printf ("Digest = ");
        MDPrint (digest);
        printf ("\nTime = %ld seconds\n", (long)(endTime-startTime));
        printf ("Speed = %ld bytes/second\n",
                (long)TEST_BLOCK_LEN * (long)TEST_BLOCK_COUNT/(endTime-startTime));
}

/* Создает сигнатуры тестовых строк и выводит результаты. */
static void MDTestSuite ()
{
        printf ("MD%d test suite:\n", MD);

        MDString ("");
        MDString ("a");
        MDString ("abc");
        MDString ("message digest");
        MDString ("abcdefghijklmnopqrstuvwxyz");
        MDString ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789");
        MDString ("12345678901234567890123456789012345678901\
                  234567890123456789012345678901234567890");
}

/* Создает и выводит сигнатуру файла. */
static void MDFile (filename)
char *filename;
{
        FILE *file;
        MD_CTX context;
        int len;
        unsigned char buffer[1024], digest[16];

        if ((file = fopen (filename, "rb")) == NULL)
                printf ("%s can't be opened\n", filename);
        else {
                MDInit (&context);
                while (len = fread (buffer, 1, 1024, file))
                        MDUpdate (&context, buffer, len);
                MDFinal (digest, &context);
                fclose (file);

                printf ("MD%d (%s) = ", MD, filename);
                MDPrint (digest);
                printf ("\n");
        }
}

/* Создает сигнатуру данных со стандартного ввода и выводит результат. */
static void MDFilter ()
{
        MD_CTX context;
        int len;
        unsigned char buffer[16], digest[16];

        MDInit (&context);
        while (len = fread (buffer, 1, 16, stdin))
                MDUpdate (&context, buffer, len);
        MDFinal (digest, &context);
```

[1] В оригинале параметры в этой строке ошибочно указаны в обратном порядке. *Прим. перев.*

```
        MDPrint (digest);
        printf ("\n");
}

/* Выводит цифровую подпись в шестнадцатеричном формате. */
static void MDPrint (digest)
unsigned char digest[16];
{
        unsigned int i;

        for (i = 0; i < 16; i++)
                printf ("%02x", digest[i]);
}
```

A.5 Тестовый набор

Тестовый набор MD5 (опция -x в командной строке драйвера) должен давать следующие результаты для MD5:

```
MD5 ("") = d41d8cd98f00b204e9800998ecf8427e
MD5 ("a") = 0cc175b9c0f1b6a831c399e269772661
MD5 ("abc") = 900150983cd24fb0d6963f7d28e17f72
MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5 ("abcdefghijklmnopqrstuvwxyz") = c3fcd3d76192e4007dfb496cca67e13b
MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") =
    d174ab98d277d9f5a5611c2c9f419d9f
MD5 ("12345678901234567890123456789012345678901234567890123456789012345678901234567890") =
    57edf4a22be3c955ac49da2e2107b67a
```

Вопросы безопасности

Уровень безопасности рассмотренного в документе алгоритма считается достаточным для реализации гибридных систем создания цифровых подписей с очень высокой защищенностью на основе MD5 и криптосистем с открытыми ключами.

Адрес автора

Ronald L. Rivest

Massachusetts Institute of Technology

Laboratory for Computer Science

NE43-324

545 Technology Square

Cambridge, MA 02139-1986

Phone: (617) 253-5880

EMail: rivest@theory.lcs.mit.edu

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru