



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Московский государственный технический университет имени  
Н. Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н. Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

## Отчёт по лабораторной работе №4 по курсу «Защита информации»

Тема Электронная подпись (RSA, MD5)

---

Студент Калашков П. А.

---

Группа ИУ7-76Б

---

Оценка (баллы)

---

Преподаватели Чиж И. С.

---

# Введение

Шифрование информации — занятие, которым человек занимался ещё до начала первого тысячелетия, занятие, позволяющее защитить информацию от посторонних лиц.

Криптографический алгоритм RSA — алгоритм, разработанный в 1977 году и положивший основу первой системе, пригодной как для шифрования, так и для цифровой подписи.

Хеширование — процесс преобразования набора данных произвольной длины в выходной набор данных установленной длины, выполняемый определённым алгоритмом.

MD5 — алгоритм хеширования, разработанный в 1991 году профессором Рональдом Л. Ривестом из Массачусетского технологического института. Он предназначен для получения последовательности длиной 128 бит из сообщений произвольной длины с целью проверки их подлинности.

**Целью данной работы** является реализация в виде программы на языке программирования C или C++, позволяющую создать и проверить электронную подпись для документа с использованием алгоритма RSA и алгоритма хеширования MD5.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) изучить криптографический алгоритм RSA и алгоритм хеширования MD5;
- 2) реализовать криптографический алгоритм RSA в виде программы, обеспечив возможности создания и проверки подлинности электронной подписи для документа с использованием алгоритма MD5;
- 3) протестировать разработанную программу, показать, что удаётся создавать и проверять электронные подписи;
- 4) описать и обосновать полученные результаты в отчёте о выполненной лабораторной работе.

# 1 Аналитическая часть

В этом разделе будут рассмотрен криптографический алгоритм RSA, алгоритм хеширования MD5, понятие электронной подписи и принципы её получения и проверки с использованием алгоритмов RSA и MD5.

## 1.1 Алгоритм RSA

Криптографический алгоритм RSA (аббревиатура от фамилий *Rivest*, *Shamir* и *Adleman*) — ассиметричный криптографический алгоритм, в основе которого лежит сложность задачи факторизации произведения двух взаимно простых чисел.

Для шифрования используется операция возведения в степень по модулю большого числа, а для дешифрации — вычисление функции Эйлера от этого большого числа за разумное время, что можно осуществить при наличии информации о разложении данного большого числа на простые множители.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. *public key*), так и закрытым ключом (англ. *private key*). В криптографической системе RSA каждый ключ состоит из пары целых чисел.

RSA ключи генерируются следующим образом:

- 1) выбираются два отличающихся друг от друга случайных простых числа  $p$  и  $q$ , лежащие в установленном диапазоне;
- 2) вычисляется их произведение  $n = p \cdot q$ , называемое модулем;
- 3) вычисляется значение функции Эйлера от числа  $n$ :  $\phi(n) = (p-1) \cdot (q-1)$ ;
- 4) выбирается целое число  $e$  ( $1 < e < \phi(n)$ ), взаимно простое со значением  $\phi(n)$ , оно называется открытой экспонентой;
- 5) вычисляется число  $d = e^{-1} \bmod(\phi(n))$ , оно называется закрытой экспонентой.

Пара  $(e, n)$  публикуются в качестве открытого ключа RSA, а пара  $(d, n)$  — в виде закрытого ключа.

Шифрование сообщения  $m$  ( $0 < m < n - 1$ ) в зашифрованное сообщение  $c$  производится по формуле  $c = E(m, k_1) = E(m, n, e) = m^e \bmod(n)$ .

Дешифрация:  $m = D(c, k_2) = D(c, n, d) = c^d \bmod(n)$

У данного принципа имеются следующие минусы:

- 1) если  $m = 0$ , то и  $c = 0$ ;
- 2) если  $m_1 = m_2$ , то и  $c_1 = c_2$ .

Из-за этого RSA используется для передачи ключей других шифров.

## 1.2 Алгоритм MD5

MD5 (англ. *Message Digest 5*) — алгоритм хеширования, предназначенный для получения последовательности длиной 128 бит, используемой для последующей проверки подлинности сообщений произвольных длин.

На вход алгоритма поступает последовательность бит произвольной длины  $L$ , хеш которой нужно найти. Алгоритм MD5 состоит из 4 следующих этапов

- 1) выравнивание потоков;
- 2) добавление длины сообщения;
- 3) инициализация буфера;
- 4) вычисления в цикле.

Выравнивание потоков представляет из себя добавление некоторого числа нулевых бит такое, чтобы новая длина последовательности  $L'$  стала сравнима с 448 по модулю 512. Выравнивание происходит в любом случае, даже если длина исходного потока уже сравнима с 448

Под добавлением длины сообщения представляет из себя добавление 64 битов в последовательность: сначала младшие 4 байта, потом старшие 4 байта. После этого длина потока станет кратной 512. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит.

После этого происходит инициализация буфера, состоящего из 4-х переменных размерностью 32 бита, начальные значения которых задаются шестнадцатеричными числами. В этих переменных будут храниться результаты промежуточных вычислений.

Далее в цикле каждый блок длиной 512 бит проходит 4 этапа вычислений по 16 раундов. Для этого блок представляется в виде массива  $X$  из 16 слов по 32 бита. Все раунды однотипны и имеют вид:  $[abcd\ k\ s\ i]$ , определяемый как  $a = b + ((a + Fun(b, c, d) + X[k] + T[i]) \ll s)$ , где  $k$  — номер 32-битного слова из текущего блока, число  $s$  задаётся отдельно для каждого раунда,  $T$  — таблица констант.

Результат вычислений хранится в переменных  $a$ ,  $b$ ,  $c$  и  $d$ .

### 1.3 Электронная подпись

Электронная (цифровая) подпись — ЭП — позволяет подтвердить авторство электронного документа. Она связана не только с автором документа, но и с самим документом (при помощи криптографических методов) и не может быть подделана при помощи обычного копирования.

Создание ЭП с использованием криптографического алгоритма RSA и алгоритма хеширования MD5 происходит следующим образом:

- 1) происходит хеширование сообщения при помощи MD5, сообщение — файл, который необходимо подписать;
- 2) происходит шифрование с использованием закрытого ключа RSA последовательности 128 бит, полученных на предыдущем этапе;
- 3) значение подписи — результат шифрования.

Проверка ЭП с использованием криптографического алгоритма RSA и алгоритма хеширования MD5 происходит следующим образом:

- 1) происходит хеширование сообщения при помощи MD5, сообщение — файл, подпись которого необходимо проверить;
- 2) происходит дешифрация подписи с использованием открытого ключа RSA;

- 3) происходит побитовая сверка значений, полученных на предыдущих этапах, если они одинаковы, подпись считается подлинной.

## Вывод

В данном разделе был рассмотрен криптографический алгоритм RSA, алгоритм хеширования MD5, а также понятие электронной подписи и принципы её получения и проверки с использованием алгоритмов RSA и MD5.

## 2 Конструкторская часть

В этом разделе будут представлены описания модулей программы, а также представлены схемы алгоритма RSA, и алгоритма хеширования MD5.

### 2.1 Сведения о модулях программы

Программа состоит из четырёх модулей:

- 1) *main.c* — файл, содержащий точку входа;
- 2) *menu.c* — файл, содержащий код меню программы;
- 3) *rsa.c* — файл, содержащий реализацию криптографического алгоритма RSA;
- 4) *md5.c* — файл, содержащий реализацию алгоритма хеширования MD5.

### 2.2 Разработка алгоритмов

На рисунках 2.1–2.4 представлены схемы алгоритма RSA, и алгоритма хеширования MD5.

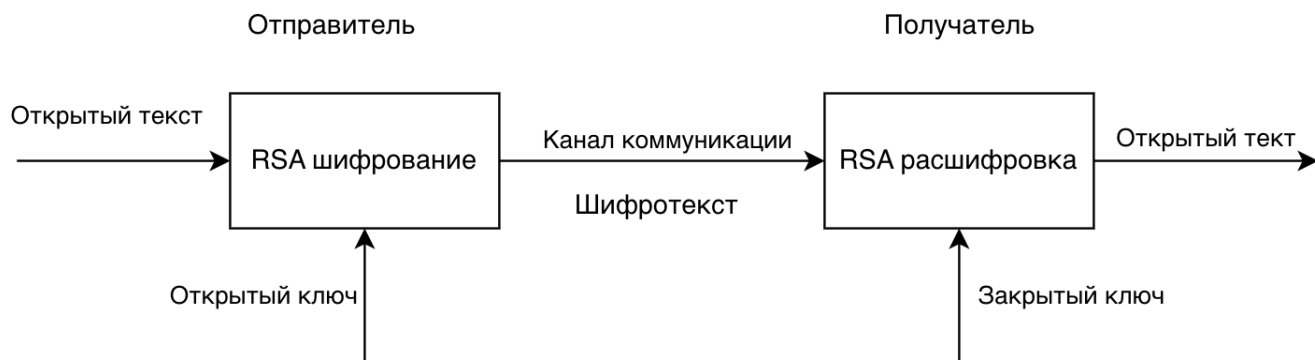


Рисунок 2.1 – Общая схема работы RSA

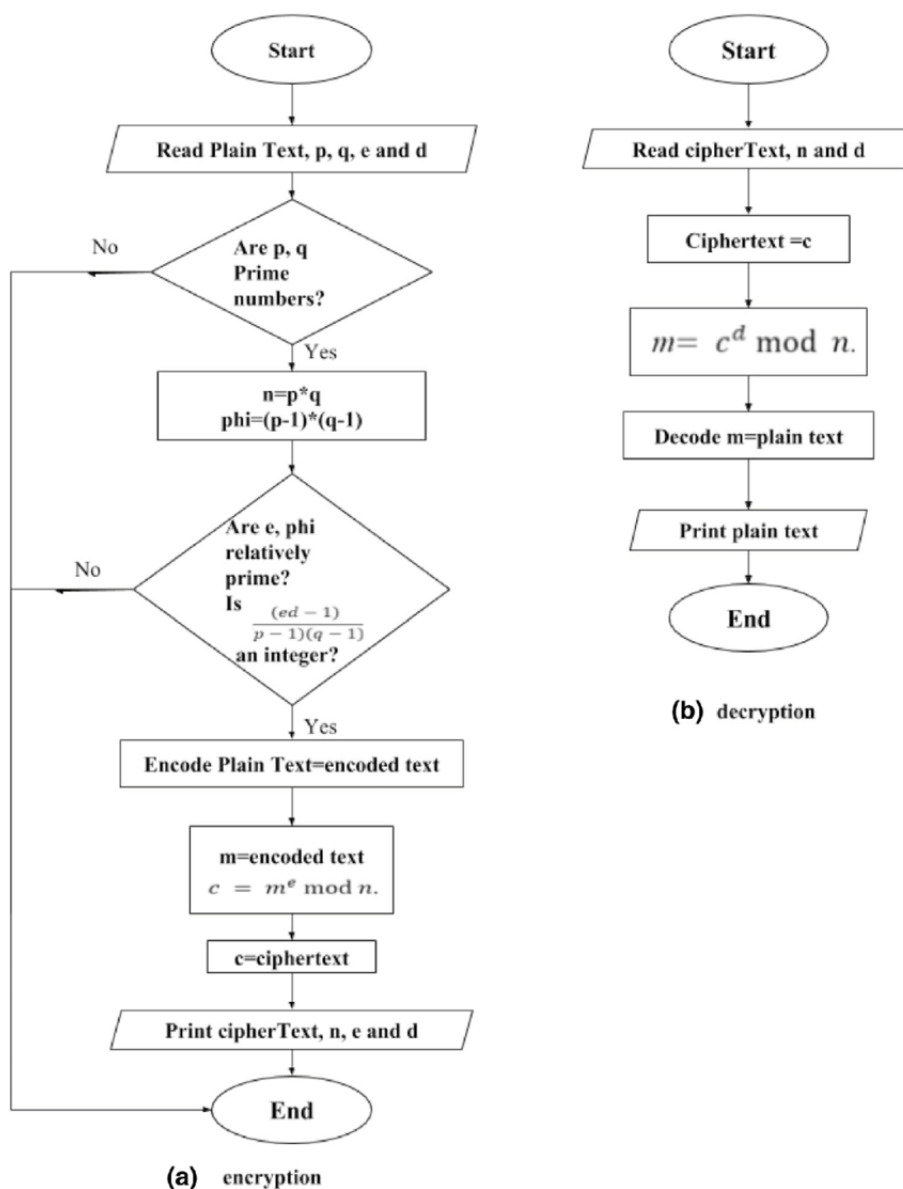


Рисунок 2.2 – RSA шифрование и дешифрация



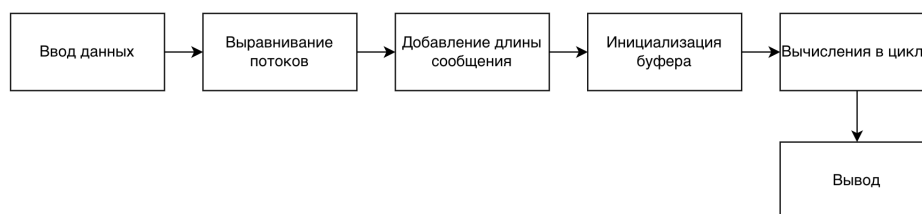


Рисунок 2.3 – Общая схема работы MD5

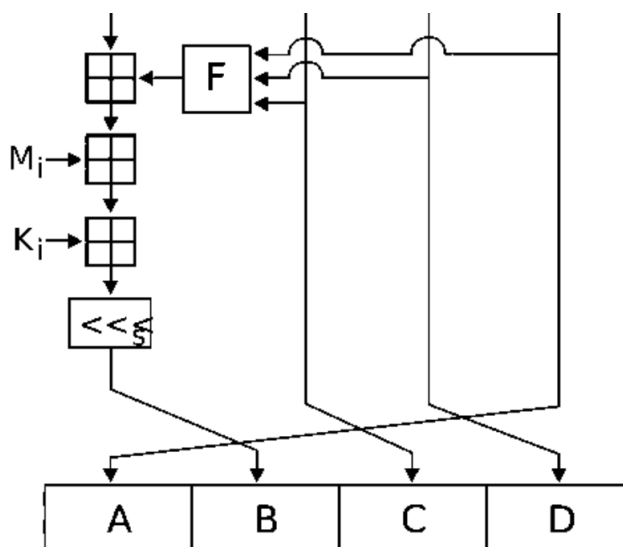


Рисунок 2.4 – Схема алгоритма вычислений в цикле

## Вывод

В данном разделе были представлены сведения о модулях программы, а также схемы алгоритмов, которые нужно реализовать: алгоритма AES, а также режима работы PCBC с зашифровкой и расшифровкой.

## 3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги реализации криптографического алгоритма RSA и алгоритма хеширования MD5, а также произведено тестирование.

### 3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *C*. Данный язык удовлетворяет поставленным критериям по средствам реализации.

### 3.2 Реализация алгоритма

В листингах 3.1–3.2 представлена реализация криптографического алгоритма RSA, на листингах 3.3–3.4 — реализация алгоритма хеширования MD5.

Листинг 3.1 – Реализация алгоритма получения ключей RSA часть 1

```
1 void generate_keys(private_key* ku, public_key* kp)
2 {
3     char buf[BUFFER_SIZE];
4     int i;
5     mpz_t phi; mpz_init(phi);
6     mpz_t tmp1; mpz_init(tmp1);
7     mpz_t tmp2; mpz_init(tmp2);
8
9     mpz_set_ui(ku->e, 3);
10
11     for(i = 0; i < BUFFER_SIZE; i++)
12         buf[i] = rand() % 0xFF;
13     buf[0] |= 0xC0;
14     buf[BUFFER_SIZE - 1] |= 0x01;
15     mpz_import(tmp1, BUFFER_SIZE, 1, sizeof(buf[0]), 0, 0, buf);
16     mpz_nextprime(ku->p, tmp1);
17     mpz_mod(tmp2, ku->p, ku->e);
```

Листинг 3.2 – Реализация алгоритма получения ключей RSA часть 2

```

1  while (!mpz_cmp_ui(tmp2, 1))
2      {
3          mpz_nextprime(ku->p, ku->p);
4          mpz_mod(tmp2, ku->p, ku->e);
5      }
6
7  do {
8      for(i = 0; i < BUFFER_SIZE; i++)
9          buf[i] = rand() % 0xFF;
10     buf[0] |= 0xC0;
11     buf[BUFFER_SIZE - 1] |= 0x01;
12     mpz_import(tmp1, (BUFFER_SIZE), 1, sizeof(buf[0]), 0, 0,
13         buf);
14     mpz_nextprime(ku->q, tmp1);
15     mpz_mod(tmp2, ku->q, ku->e);
16     while (!mpz_cmp_ui(tmp2, 1))
17     {
18         mpz_nextprime(ku->q, ku->q);
19         mpz_mod(tmp2, ku->q, ku->e);
20     } while (mpz_cmp(ku->p, ku->q) == 0);
21
22     mpz_mul(ku->n, ku->p, ku->q);
23
24     mpz_sub_ui(tmp1, ku->p, 1);
25     mpz_sub_ui(tmp2, ku->q, 1);
26     mpz_mul(phi, tmp1, tmp2);
27
28     if (mpz_invert(ku->d, ku->e, phi) == 0)
29     {
30         mpz_gcd(tmp1, ku->e, phi);
31         printf("gcd(e, phi) = [%s]\n", mpz_get_str(NULL, 16, tmp1));
32         printf("Invert failed\n");
33     }
34
35     mpz_set(kp->e, ku->e);
36     mpz_set(kp->n, ku->n);

```

### Листинг 3.3 – Реализация алгоритма хеширования MD5 часть 1

```

1 void md5(const uint8_t *initial_msg, size_t initial_len, uint8_t
   *digest) {
2     uint32_t h0, h1, h2, h3;
3     uint8_t *msg = NULL;
4     size_t new_len, offset;
5     uint32_t w[16];
6     uint32_t a, b, c, d, i, f, g, temp;
7     h0 = 0x67452301;
8     h1 = 0xefcdab89;
9     h2 = 0x98badcfe;
10    h3 = 0x10325476;
11    for (new_len = initial_len + 1; new_len % (512/8) != 448/8;
        new_len++);
12    msg = (uint8_t*)malloc(new_len + 8);
13    memcpy(msg, initial_msg, initial_len);
14    msg[initial_len] = 0x80; // append the "1" bit; most
        significant bit is "first"
15    for (offset = initial_len + 1; offset < new_len; offset++)
16        msg[offset] = 0; // append "0" bits
17    to_bytes(initial_len*8, msg + new_len);
18    to_bytes(initial_len>>29, msg + new_len + 4);
19    for(offset=0; offset<new_len; offset += (512/8)) {
20        for (i = 0; i < 16; i++)
21            w[i] = to_int32(msg + offset + i*4);
22        a = h0;
23        b = h1;
24        c = h2;
25        d = h3;
26        for(i = 0; i<64; i++) {
27            if (i < 16) {
28                f = (b & c) | ((~b) & d);
29                g = i;
30            } else if (i < 32) {
31                f = (d & b) | ((~d) & c);
32                g = (5*i + 1) % 16;
33            } else if (i < 48) {
34                f = b ^ c ^ d;
35                g = (3*i + 5) % 16;

```

### Листинг 3.4 – Реализация алгоритма хеширования MD5 часть 2

```
1         } else {
2             f = c ^ (b | (~d));
3             g = (7*i) % 16;
4         }
5         temp = d;
6         d = c;
7         c = b;
8         b = b + LEFTROTATE((a + f + k[i] + w[g]), r[i]);
9         a = temp;
10    }
11    h0 += a;
12    h1 += b;
13    h2 += c;
14    h3 += d;
15 }
16 free(msg);
17 to_bytes(h0, digest);
18 to_bytes(h1, digest + 4);
19 to_bytes(h2, digest + 8);
20 to_bytes(h3, digest + 12);
21 }
```

## 3.3 Тестирование

Тестирование разработанной программы производилось следующим образом: выбирались случайные значения содержимого файла длиной  $n$ . Для данного содержимого файла составлялся подпись, после чего осуществлялась её проверка. Данная процедура повторялась  $n$  раз для значений  $n$  от 1 до 100.

Таблица 3.1 – Функциональные тесты

Длина, байты	Шифруемое значение	Результат работы
8	12345678	Сообщение об ошибке
16	1234567812345678	cad29cf5b295a4bf 5905026c48d83c5
32	1234567812345678 1234567812345678	cad29cf5b295a4bf 590d5026c48d83c5 9ab00f0ae0135012 710c4ba8595b138c

## Вывод

В данном разделе были рассмотрены описания модулей программы, а также представлены схемы алгоритма RSA, и алгоритма хеширования MD5 и проведено функциональное тестирование.

# Заключение

В результате лабораторной работы была реализована программа, позволяющая создать и проверить электронную подпись для документа с использованием алгоритма RSA и алгоритма хеширования MD5.

Были выполнены следующие задачи:

- 1) изучен криптографический алгоритм RSA и алгоритм хеширования MD5;
- 2) реализован криптографический алгоритм RSA в виде программы, обеспечив возможности создания и проверки подлинности электронной подписи для документа с использованием алгоритма MD5;
- 3) протестирована разработанная программа;
- 4) описаны и обоснованы полученные результаты в отчёте о выполненной лабораторной работе.