*Выполнил студент ПК-12-2, Бекленищев Владислав*

# Описание программы

## Общие сведения

Программа “Digital signing” является результатом выполнения лабораторной работы для курса “Методы защиты информации” и создана только в учебных целях, для демонстрации зашифровывания / расшифровывания различных текстовых сообщений.

Для запуска программы необходимо следующее программное обеспечение:

* Операционная система Windows 7, 8, 8.1;
* Библиотека .NET Framework версии 4.5.

Данная программа была написана с помощью средств языка C# на платформе .NET Framework в среде Microsoft Visual Studio 2012.

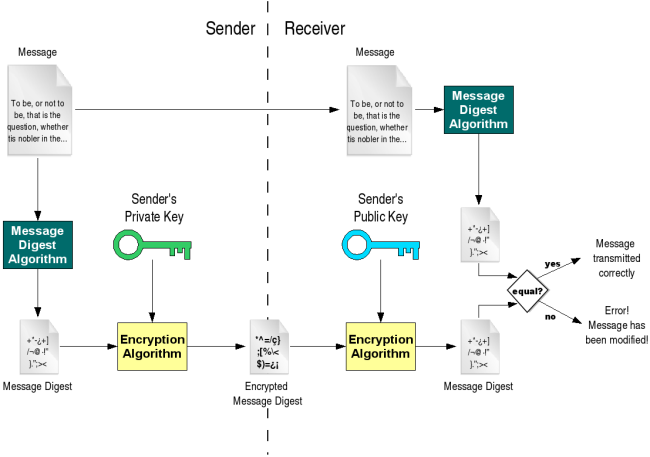
## Функциональное назначение

Приложение предназначено для подписи текстовых сообщений с помощью алгоритма хеширования SHA1 и ассиметрического алгоритма шифрования RSA.

## Описание логической структуры

Как уже было сказано выше, в основе программы “ Digital signing ” лежит алгоритм хеширования SHA1 и алгоритм ассиметрического шифрования шифрования RSA.

Общую схему шифрования можно подать в таком виде:



### Алгоритм хеширования SHA1

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента.

Псевдокод алгоритма SHA-1 следующий:

*Замечание: Все используемые переменные 32 бита.*  
  
*Инициализация переменных:*  
h0 = 0x67452301  
h1 = 0xEFCDAB89  
h2 = 0x98BADCFE  
h3 = 0x10325476  
h4 = 0xC3D2E1F0  
  
*Предварительная обработка:*  
Присоединяем бит '1' к сообщению  
Присоединяем k битов '0', где k наименьшее число ≥ 0 такое, что длина получившегося сообщения  
(в битах) [сравнима по модулю](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BF%D0%BE_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8E_%D0%BD%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D1%87%D0%B8%D1%81%D0%BB%D0%B0) 512 с 448 (length mod 512 == 448)  
Добавляем длину исходного сообщения (до предварительной обработки) как целое 64-битное  
[Big-endian](https://ru.wikipedia.org/wiki/Big-endian) число, в *битах*.  
  
*В процессе сообщение разбивается последовательно по 512 бит:*  
**for** перебираем все такие части  
 разбиваем этот кусок на 16 частей, слов по 32-бита w[i], 0 <= i <= 15  
  
 *16 слов по 32-бита дополняются до 80 32-битовых слов:*  
 **for** i **from** 16 **to** 79  
 w[i] = (w[i-3] **xor** w[i-8] **xor** w[i-14] **xor** w[i-16]) [**циклический сдвиг влево**](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B4%D0%B2%D0%B8%D0%B3) 1  
  
 *Инициализация хеш-значений этой части:*  
 a = h0  
 b = h1  
 c = h2  
 d = h3  
 e = h4  
  
 *Основной цикл:*  
 **for** i **from** 0 **to** 79  
 **if** 0 ≤ i ≤ 19 **then**  
 f = (b **and** c) **or** ((**not** b) **and** d)  
 k = 0x5A827999  
 **else if** 20 ≤ i ≤ 39 **then**  
 f = b **xor** c **xor** d  
 k = 0x6ED9EBA1  
 **else if** 40 ≤ i ≤ 59 **then**  
 f = (b **and** c) **or** (b **and** d) **or** (c **and** d)  
 k = 0x8F1BBCDC  
 **else if** 60 ≤ i ≤ 79 **then**  
 f = b **xor** c **xor** d  
 k = 0xCA62C1D6  
  
 temp = (a **leftrotate** 5) + f + e + k + w[i]  
 e = d  
 d = c  
 c = b **leftrotate** 30  
 b = a  
 a = temp  
  
 *Добавляем хеш-значение этой части к результату:*  
 h0 = h0 + a  
 h1 = h1 + b   
 h2 = h2 + c  
 h3 = h3 + d  
 h4 = h4 + e  
  
*Итоговое хеш-значение:*  
digest = hash = h0 **append** h1 **append** h2 **append** h3 **append** h4

Вместо оригинальной формулировки FIPS PUB 180-1 приведены следующие эквивалентные выражения и могут быть использованы на компьютере f в главном цикле:

(0 ≤ i ≤ 19): f = d **xor** (b **and** (c **xor** d)) *(альтернатива 1)*  
(0 ≤ i ≤ 19): f = (b **and** c) **xor** ((**not** b) **and** d) *(альтернатива 2)*  
(0 ≤ i ≤ 19): f = (b **and** c) + ((**not** b) **and** d) *(альтернатива 3)*  
   
(40 ≤ i ≤ 59): f = (b **and** c) **or** (d **and** (b **or** c)) *(альтернатива 1)*  
(40 ≤ i ≤ 59): f = (b **and** c) **or** (d **and** (b **xor** c)) *(альтернатива 2)*  
(40 ≤ i ≤ 59): f = (b **and** c) + (d **and** (b **xor** c)) *(альтернатива 3)*  
(40 ≤ i ≤ 59): f = (b **and** c) **xor** (b **and** d) **xor** (c **and** d) *(альтернатива 4)*

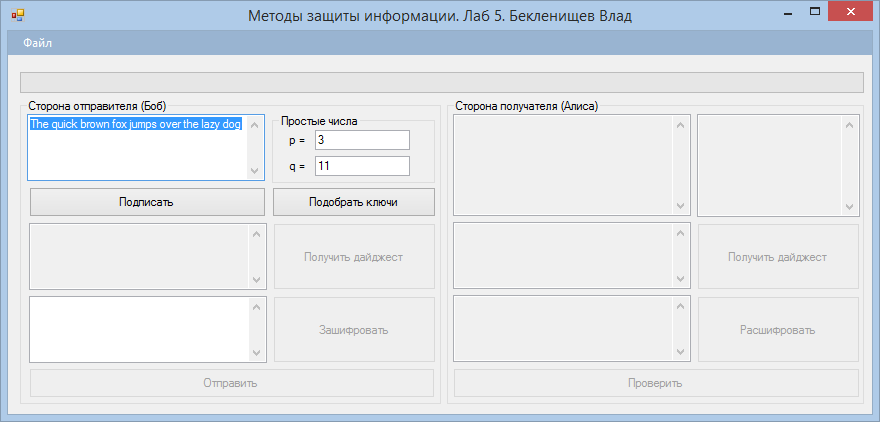
### Этапы шифрования

1. Подобрать открытый и закрытый ключи с помощью ассиметрического алгоритма шифрования. В программе используется RSA алгоритм.
2. Подписать исходное сообщение путём добавления в конец сообщение подписи: имя создателя подписи, время, дата и тд.
3. Хешировать подписанное исходное сообщение с помощью алгоритма SHA1.
4. Полученный в 3 пункте дайджест зашифровуем с помощью закрытого ключа.
5. Отправляем получателю сообщение, зашифрованный дайджест и открытый ключ.
6. Получатель генерирует свой дайджест исходного сообщения с помощью SHA1.
7. Получатель расшифровует зашифрованный дайджест (полученный в пункте 4) с помощью открытого ключа.
8. Полученные дайджесты из пунктов 6 и 7 сравниваются. Если они идентичны, то сообщение передано корректно. Иначе – сообщение изменили в ходе передачи.

## Вызов и загрузка

Так как программа является приложением под операционные системы Windows 7, 8, 8.1, 10 её можно запустить либо из меню “Пуск” либо же с помощью ярлыка на рабочем столе.

После запуска появится главное окно приложения:



## Входные данные

Входными данными являются:

1. Текстовое сообщение, которое можно подписать с помощью кнопки «Подписать».
2. Простые числа p и q. После их ввода нужно нажать кнопку «Подобрать ключи» чтобы RSA смог подобрать открытый и закрытый ключи.
3. После получения зашифрованного дайджеста (после нажатия кнопки «Зашифровать») есть возможность изменить зашифрованное сообщение. Это сделано только для того чтобы можно было продемонстрировать ошибку верификации двух дайджестов в области получателя после нажатия на кнопку «Проверить».

Также программа позволяет считать с файла текстовое сообщение для передачи получателю.

## Выходные данные

Результатом работы программы будет:

1. Дайджесты на стороне получателя и отправителя.
2. Зашифрованные дайджесты на стороне получателя и отправителя.
3. Расшифрованный дайджест на стороне получателя. Собственно дайджест полученный с помощью алгоритма шифрования SHA1 проверяется с дайджестом полученным в результате дешифровки с помощью открытого ключа на стороне получателя.