

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине: «Теория информации»

на тему: «Электронная цифровая подпись»

ВЫПОЛНИЛ

студент группы  
Яковлев Н.А.

ПРОВЕРИЛ

ст. преподаватель  
Захарова И.Ю.

Новополоцк, 2020 г.

**Цель работы:** изучить основные понятия, связанные с электронной цифровой подписью. Разобрать и научиться использовать на практике алгоритм безопасного хеширования SHA-1, а также алгоритм цифровой подписи RSA.

### Теоретическая часть

Алгоритм безопасного хеширования SHA-1 был опубликован в 1995 году в качестве замены использовавшегося до этого алгоритма хеширования SHA-0, в котором была обнаружена уязвимость.

Для сообщения произвольной длины  $l$ , не превышающей  $2^{64}$  бит, алгоритм SHA-1 формирует 160-битный хеш-образ.

Процедура формирования хеш-образа состоит из следующих шагов.

1. Весь исходный текст разбивается на блоки по 512 бит. В случае если длина исходного текста не кратна 512 битам, производится его выравнивание за счёт добавления в конец бита со значением 1,  $m$  нулевых битов и 64-битного представления значения длины исходного сообщения (рис. 1).

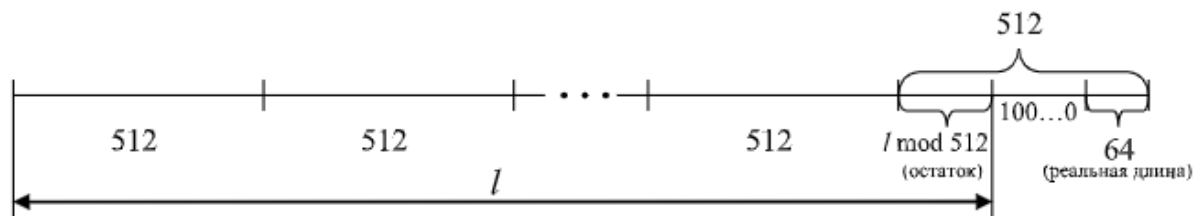


Рис. 1 . Выравнивание исходного сообщения

2. Инициализируется пять 32-битных рабочих переменных A, B, C, D, E

3. Выполняется обработка очередных 512 бит исходного текста. Для этого значения переменных A, B, C, D, E копируются в переменные a, b, c, d, e и далее для  $t$  от 1 до 80 выполняется преобразование значений данных переменных по схеме, изображенной на рис. 2 .

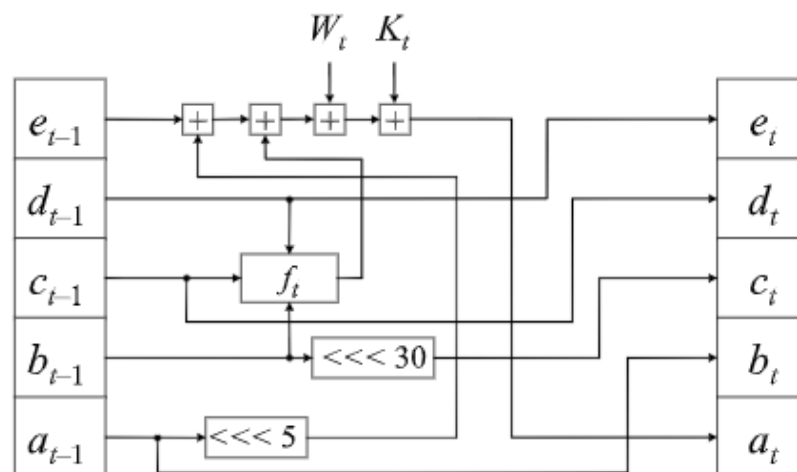


Рис. 2 . Схема итерации алгоритма SHA-1

4. Значения переменных  $a, b, c, d, e$  независимо друг от друга складываются по модулю  $2^{32}$  со значениями переменных  $A, B, C, D, E$ , в которые затем и помещаются полученные результаты.

5. Шаги 3–4 выполняются до тех пор, пока не будет обработан весь текст.

После обработки последнего блока текста значение хеш-образа формируется как ABCDE.

### Алгоритм цифровой подписи RSA

Математическая схема электронной цифровой подписи по алгоритму RSA была предложена в 1977 году сотрудниками Массачусетского технологического института США. Данная система цифровой подписи стала первым практическим решением задачи подписи электронных документов при помощи криптосистем с открытым ключом. Процедура вычисления цифровой подписи в данной системе использует криптографическое преобразование по алгоритму RSA.

В соответствии с данной системой цифровой подписи, субъект, желающий пересылать подписанные им документы, должен сформировать два ключа алгоритма RSA: открытый и закрытый.

Пару значений  $(K_O, r)$ , которая является открытым ключом подписи, отправитель передаёт всем возможным получателям его сообщений. Именно эти значения будут использоваться для проверки подлинности и принадлежности отправителю полученных от него сообщений.

Значение  $K_C$  сохраняется отправителем в секрете. Данное значение вместе с модулем  $r$  является секретным ключом, который будет использоваться отправителем для постановки подписей под своими сообщениями. Схема использования алгоритма цифровой подписи на базе RSA для обмена двух абонентов подписанными сообщениями показана на рис. 3.

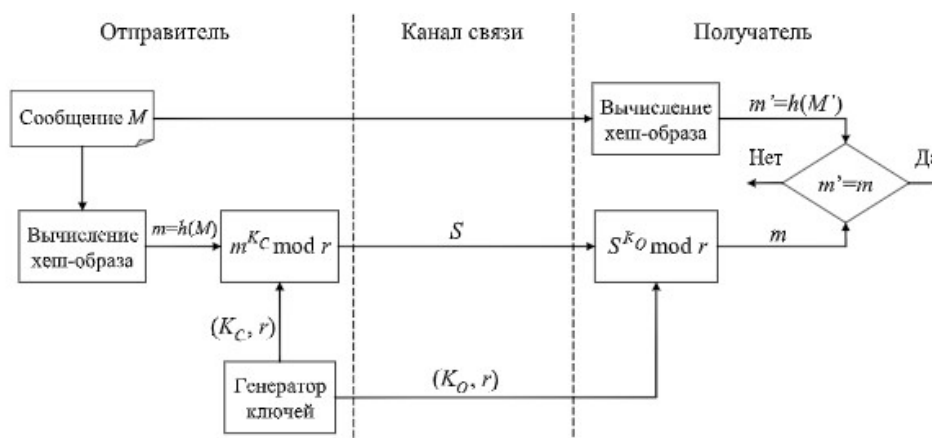


Рис. 3. Схема использования цифровой подписи на базе RSA

Допустим, что получатель уже располагает открытым ключом подписи отправителя. Процедура подписи отправителем сообщения  $M$  будет состоять из следующих шагов.

1. Отправитель сжимает сообщение  $M$  при помощи криптографической хеш-

функции  $h$  в целое число  $m = h(M)$ .

2. Отправитель вычисляет значение цифровой подписи  $S$  для сообщения  $M$  на

основе ранее полученного значения хеш-образа  $m$  и значения своего закрытого

(секретного) ключа подписи  $K_C$ . Для этого используется преобразование, аналогичное

преобразованию, выполняемому при шифровании по алгоритму RSA:  $S = m K_C \bmod r$ .

Пара  $(M, S)$ , представляющая собой подписанное отправителем сообщение,

передаётся получателю. Сформировать подпись  $S$  мог только обладатель закрытого ключа  $K_C$ .

Процедура проверки получателем подлинности сообщения и принадлежности его

отправителю состоит из следующих шагов.

1. Получатель сжимает полученное сообщение  $M'$  при помощи криптографической хеш-функции  $h$ , идентичной той, которая была использована

отправителем, в целое число  $m'$ .

2. Получатель выполняет расшифрование открытым ключом  $K_O$  О

дайджеста  $m$  оригинального сообщения, преобразуя значение подписи  $S$  по алгоритму

RSA:  $m = S K_O \bmod r$ .

3. Получатель сравнивает полученные значения  $m'$  и  $m$ . Если данные значения

совпадают, т. е.

$$S K_O \bmod r = h(M')$$

то получатель признает полученное сообщение подлинным и принадлежащим отправителю.

## Порядок выполнения работы

1. Изучить теоретический материал.
2. Реализовать алгоритм вычисления хеш-функции SHA-1 для файла с произвольным размером и содержимым.
3. Реализовать программное средство, выполняющее генерацию и проверку ЭЦП файла с произвольным содержимым на базе алгоритма RSA с использованием для вычисления хеш-функции ранее реализованного алгоритма SHA-1.

## Практическая часть

Для тестирования возьмем элементарное сообщение «Hello World». Для вычисления хеш функции я использую метод `getInstance()`, класса `MessageDigest`. Код метода `getInstance()` приведен в приложении. Код приведен на рисунке 5.

```
public static void main(String[] args) throws NoSuchAlgorithmException {  
    // write your code here  
    String text = "Hello World";  
    System.out.println("Исходный текст : " + text);  
    MessageDigest sha1 = MessageDigest.getInstance("SHA-1");  
    byte[] bytes = sha1.digest(text.getBytes());  
    StringBuilder stringBuilder = new StringBuilder();  
    for (byte b : bytes){  
        stringBuilder.append(String.format("%02X ", b)); //конвертирую в 16-ричку  
    }  
    System.out.print("Хеш-код : " + stringBuilder.toString());  
}
```

Рис 5. - метод `main()`

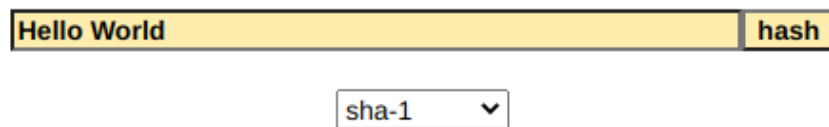
Исходное сообщение конвертирую в массив байт. Далее через цикл `foreach` перебираем массив и конвертируем каждый его элемент в 16-ричную систему исчисления. Вывод представлен на рисунке 6.

```
Исходный текст : Hello World  
Хеш-код : 0A 4D 55 A8 D7 78 E5 02 2F AB 70 19 77 C5 D8 40 BB C4 86 D0  
Process finished with exit code 0
```

Рис 6. - результат выполнения программы

Проверим корректность работы кода, через онлайн кодер SHA-1 :

# SHA1 and other hash functions online generator



Hello World hash

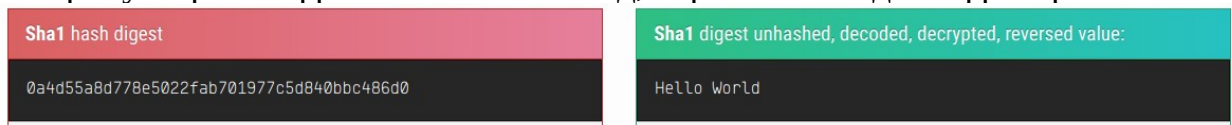
sha-1

**Result for sha1:** 0a4d55a8d778e5022fab701977c5d840bbc486d0

Рис. 7 — результат шифрования через онлайн кодер

Сравнив результаты из рисунков 6 и 7 можно сделать вывод, что алгоритм работает корректно.

Попробуем расшифровать наш хеш-код, через онлайн дешифратор :



Sha1 hash digest

0a4d55a8d778e5022fab701977c5d840bbc486d0

Sha1 digest unhashed, decoded, decrypted, reversed value:

Hello World

Рис.8 — результат онлайн дешифратора

Исходя из рисунка 8, можно сделать вывод, что примитивные сообщения возможно расшифровать без труда. Соответственно, для сохранения данных требуются придумывать нестандартные пароли, т.к пароли хэшируются. Если пароль состоит из сложных и нестандартных символов, злоумышленнику будет тяжелее взломать ваш хэш-код, так как пароли хранятся в виде хэш-кода.

**Вывод :** в ходе данной лабораторной работы был реализован алгоритм вычисления хэш-функции SHA-1. Были произведены проверки, через онлайн кодеры и декодере. Алгоритм работает корректно.

## Контрольные вопросы

**What is the purpose of the hash function and what are the requirements for the hash functions used for staging EDS?**

The hashing function can be used to detect changes in a message, i.e. it can serve to form a cryptographic checksum (also called change detection code or message authentication code). In this capacity, the hash function is used to control the integrity of a message when forming and checking the EDS.

The hashing function must have the following properties:

1. The hash function can be applied to an argument of any size.
2. The output value of the hash function has a fixed size.
- 3 The hash function  $h(x)$  can be simply calculated for any  $x$ . The speed of hash function calculation must be such that the speed of EDS generation and verification when using the hash function is much higher than when using the message itself.

The hash function must be sensitive to all kinds of changes in  $M$  text, such as inserts, ejections, permutations, etc.

5. The hash function must be unidirectional, i.e. it must have the property of irreversibility, in other words, the task of selecting the document  $M'$  that has the required value of the hash function must be computationally unsolvable.
6. The probability that the hash function values of two different documents (regardless of their lengths) will coincide must be negligibly small; i.e. for any fixed  $x$ , it is computationally impossible to find  $x' \neq x$ , such that  $h(x') = h(x)$ .

**Describe EDS staging and verification procedures. What information is contained in an EDS?**

EDS system includes two procedures: 1) procedure of signing; 2) procedure of signature verification. The secret key of the message sender is used in the signature creation procedure, and the public key of the sender is used in the signature verification procedure.

An electronic digital signature (EDS) is an electronic document detail designed to protect a given electronic document from forgery, obtained as a result of cryptographic transformation of information using the electronic digital signature's private key and allowing to identify the owner.

**On what principles is the cryptographic stability of modern EDS algorithms based?**

Many modern methods of protective transformations can be classified into four large groups: permutations, substitutions, additive and combined methods. Methods of permutations and substitutions are usually characterized by short key lengths, and the reliability of their protection is determined by the complexity of transformation algorithms. The additive methods are characterized by simple transformation algorithms, and their cryptostability is based on increasing the key length.

### Приложение

```
public static MessageDigest getInstance(String algorithm,
                                       Provider provider)
    throws NoSuchAlgorithmException
{
    if (provider == null)
        throw new IllegalArgumentException("missing provider");
    Object[] objs = Security.getImpl(algorithm, "MessageDigest", provider);
    if (objs[0] instanceof MessageDigest) {
        MessageDigest md = (MessageDigest)objs[0];
        md.provider = (Provider)objs[1];
        return md;
    } else {
        MessageDigest delegate =
            new Delegate((MessageDigestSpi)objs[0], algorithm);
        delegate.provider = (Provider)objs[1];
        return delegate;
    }
}
```

Метод getInstance() класса MessageDigest.