****

**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Институт № 3**

**Кафедра 311**

**Информационная безопасность**

**Лабораторная работа № 7**

**«Шифрование RSA»**

**Выполнил студент  
Максимов Евгений Викторович**

**Группа М3З-302-БК**

**Дата 07.06.2022 г.**

**Принял преподаватель  
Кос Оксана Игоревна**

Оглавление

[Цель лабораторной работы 3](#_Toc89796754)

[Глава 1. Принцип Шифра RSA. 4](#_Toc89796755)

[Глава 2. Алгоритм создания открытого и секретного ключей 5](#_Toc89796756)

[Глава 4. Шифрование и расшифровка 6](#_Toc89796757)

[Глава 5. Шифрование и расшифровка сеансовым ключом 7](#_Toc89796758)

[Глава 3. Шифрование текста 8](#_Toc89796759)

[Итоги лабораторной работы 10](#_Toc89796760)

[Список литературы 11](#_Toc89796761)

# Цель лабораторной работы

Изучить:

* 1. Принцип Шифра RSA;
  2. Зашифровать текст с помощью Шифра RSA;
  3. Расшифровать зашифрованный текст.

# Глава 1. Принцип Шифра RSA.

**RSA** (аббревиатура от фамилий Rivest, Shamir и Adleman) — криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования (обратной операции) за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для чего необходимо знать разложение числа на простые множители.

# Глава 2. Алгоритм создания открытого и секретного ключей

1) выбираются два различных случайных простых числа p и q заданного размера (например, 1024 бита каждое);

2) вычисляется их произведение n = p \* q, которое называется модулем;

3) вычисляется значение функции Эйлера (мультипликативная арифметическая функция, значение которой равно количеству натуральных чисел, не превышающих n и взаимно простых с ним) от числа n:

4) выбирается целое число взаимно простое со значением функции ;

число e называется открытой экспонентой;

обычно в качестве e берут простые числа, содержащие небольшое количество единичных бит в двоичной записи, например, простые из чисел Ферма: 17, 257 или 65537, так как в этом случае время, необходимое для шифрования с использованием быстрого возведения в степень, будет меньше;

слишком малые значения e, например 3, потенциально могут ослабить безопасность схемы RSA;

5) вычисляется число d, мультипликативно обратное к числу e по модулю , то есть число, удовлетворяющее сравнению:

(число d называется секретной экспонентой; обычно оно вычисляется при помощи расширенного алгоритма Евклида);

6) пара (e, n) публикуется в качестве открытого ключа RSA;

7) пара (d, n) играет роль закрытого ключа RSA и держится в секрете.

# Глава 4. Шифрование и расшифровка

**Алгоритм шифрования**:

* Взять открытый ключ (e, n)
* Взять открытый текст m
* Зашифровать сообщение с использованием открытого ключа:

**Алгоритм расшифрования:**

* Принять зашифрованное сообщение c
* Взять свой закрытый ключ (d, n)
* Применить закрытый ключ для расшифрования сообщения:

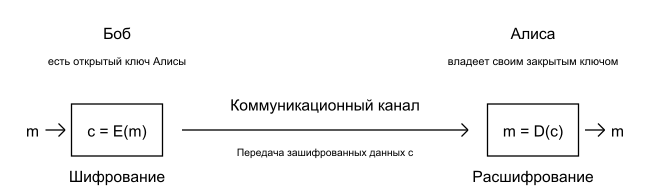


Рисунок 4.1 – Схема шифрования RSA

\* Данная схема на практике не используется, т.к. является практически не надёжной.

# Глава 5. Шифрование и расшифровка сеансовым ключом

**Алгоритм шифрования:**

* Взять открытый ключ (e, n)
* Создать случайный сеансовый ключ m
* Зашифровать сеансовый ключ с использованием открытого ключа:
* Зашифровать сообщение MA с помощью сеансового ключа симметричным алгоритмом:

**Алгоритм расшифровки:**

* Принять зашифрованный сеансовый ключ Боба c
* Взять свой закрытый ключ (d, n)
* Применить закрытый ключ для расшифровывания сеансового ключа:
* Расшифровать сообщение C с помощью сеансового ключа симметричным алгоритмом:

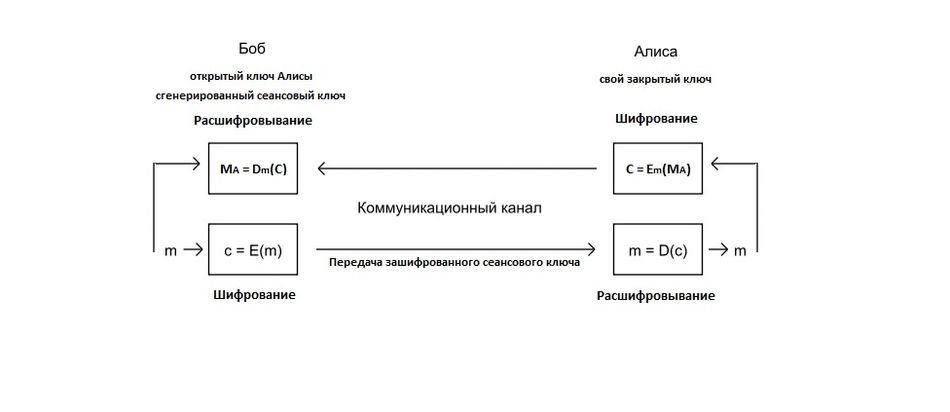


Рисунок 5.1 – Шифрование сеансовым ключом

# Глава 3. Шифрование текста

**Задача**: зашифровать текст алгоритмом RSA

**Исходный текст:** «Съешь еще этих мягких форанцузских булочек да выпей чаю»

**Зашифрованный текст:** «ZSybpaV09pleWLxzi5VKatN9Ye/jtfurLqOap3UXXIK2chhIksU+YDrlEHLKaRlIB3TmBDq6/y7Pn5Z+elIljycGJygPhYwlLPk5DQi4pKExYagCEot6rPe5zSBGmlF8z3M3ZngCMn42j6dyV3EdMg==»

**Задача:** расшифровать текст алготимом RSA

**Зашифрованный текст:** «ZSybpaV09pleWLxzi5VKatN9Ye/jtfurLqOap3UXXIK2chhIksU+YDrlEHLKaRlIB3TmBDq6/y7Pn5Z+elIljycGJygPhYwlLPk5DQi4pKExYagCEot6rPe5zSBGmlF8z3M3ZngCMn42j6dyV3EdMg==»

**Расшифрованный текст:** «Съешь еще этих мягких форанцузских булочек да выпей чаю»

**Листинг программы:**

*Метод генерации ключей RSA и сеансового ключа*

private static void generateKeys() throws Exception {  
 KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");  
 keyPairGenerator.initialize(2048); // Размер ключей будет 2048 бит  
 KeyPair keyPair = keyPairGenerator.generateKeyPair();  
 privateKey = keyPair.getPrivate();  
 publicKey = keyPair.getPublic();  
  
 KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");  
 keyGenerator.init(128); // Размер AES ключа - 128 бит  
 aesKey = keyGenerator.generateKey();  
}

*Методы шифровки и расшифровки*

public static void main(String[] args) throws Exception {  
 RSA = Cipher.getInstance("RSA");  
 AES = Cipher.getInstance("AES");  
 generateKeys();  
  
 byte[] encryptedMessageBytes = encrypt();  
 // Шифруем AES ключ  
 RSA.init(Cipher.PUBLIC\_KEY, publicKey);  
 byte[] encryptedAesKey = RSA.doFinal(aesKey.getEncoded());  
  
 RSA.init(Cipher.PRIVATE\_KEY, privateKey);  
 byte[] decryptedAesKey = RSA.doFinal(encryptedAesKey);  
 String decryptedMessage = decrypt(encryptedMessageBytes, decryptedAesKey);  
  
 System.out.println("Исходная строка: " + SOURCE\_STRING);  
 System.out.println("зашифрованная строка: " + Base64.getEncoder().encodeToString(encryptedMessageBytes));  
 System.out.println("Расшифрованная строка: " + decryptedMessage);  
  
}  
  
private static String decrypt(byte[] encryptedMessageBytes, byte[] decryptedKeyBytes) throws Exception {  
 SecretKey originalAesKey = new SecretKeySpec(decryptedKeyBytes, 0, decryptedKeyBytes.length, "AES");  
 AES.init(Cipher.DECRYPT\_MODE, originalAesKey);  
 byte[] decryptedBytes = AES.doFinal(encryptedMessageBytes);  
  
 return new String(decryptedBytes, UTF\_8);  
}  
  
private static byte[] encrypt() throws Exception {  
 AES.init(Cipher.ENCRYPT\_MODE, aesKey);  
 byte[] sourceStringBytes = SOURCE\_STRING.getBytes(UTF\_8);  
  
 return AES.doFinal(sourceStringBytes);  
}

# Итоги лабораторной работы

Мы научились шифровать, расшифровывать текст с помощью алгоритма RSA.

# Список литературы

1. [[Wikipedia.org](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F)] – Шифр RSA;