Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Практическое задание №4.2**

**Тема «Криптографическая защита информации»**

Руководитель: Ржеутская Н. В.

Выполнил:

Студент 2 курса 1 группы ФИТ

Немкович Анастасия Вадимовна

Минск 2023

**Цель**

Овладение основными криптографическими алгоритмами асимметричного шифрования.

**Ход работы**

**1.RSA-шифрование**

RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи **факторизации больших целых чисел.** Его криптостойкость и следующая отсюда всюду применимость в современных системах основывается на сложности разложения на множители больших чисел, а именно - на исключительной трудности задачи определить секретный ключ на основании открытого, так как для этого потребуется решить задачу о существовании делителей целого числа.

Весь алгоритм приведен в таблице:

|  |  |  |
| --- | --- | --- |
| Этап | Описание операции | Результат операции |
| Генерация ключей | Выбрать два простых различных числа | p=3557,  q=2579 |
| Вычислить модуль (произведение) | n = p \cdot q = 3557 \cdot 2579 = 9173503 |
| Вычислить функцию Эйлера | \varphi(n) = (p-1) (q-1) = 9167368 |
| Выбрать открытую экспоненту | e = 3 |
| Вычислить секретную экспоненту | d = e^{-1} \mod \varphi(n)  d = 6111579 |
| Опубликовать открытый ключ | \{e, n\} = \{3,9173503 \} |
| Сохранить закрытый ключ | \{d, n\} = \{6111579, 9173503 \} |
| Шифрование | Выбрать текст для зашифровки | m = 111111 |
| Вычислить шифротекст | \begin{align} c &= E(m) \\  &= m^e \mod n \\  &= 111111^3   \mod 9173503 \\  &= 4051753 \end{align} |
| Расшифрование | Вычислить исходное сообщение | \begin{align} m &= D(c) = \\   &= c^d \mod n \\   &= 4051753^{6111579} \mod 9173503 \\   &= 111111 \end{align} |

*Шаг первый. Подготовка ключей*

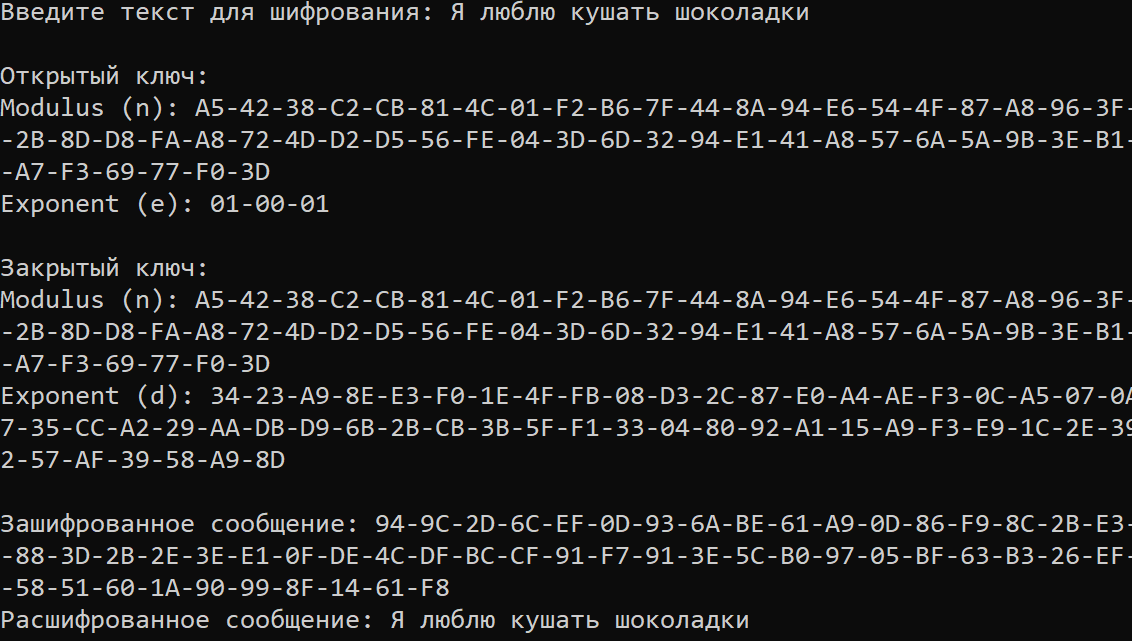
Процесс начинается с выбора двух простых чисел, которые будем обозначать как p и q. Для данной иллюстрации, возьмем p=3 и q=7. Затем мы вычисляем модуль n, который равен произведению p и q, то есть n = p × q = 3 × 7 = 21. Далее, вычисляем функцию Эйлера φ, которая равна произведению (p-1) и (q-1), таким образом, φ = (p-1) × (q-1) = 2 × 6 = 12. Теперь мы выбираем открытый ключ, обозначаемый как e. Это число должно удовлетворять нескольким критериям: (i) оно должно быть простым числом; (ii) оно должно быть меньше φ, поэтому доступными вариантами являются 3, 5, 7, 11; (iii) оно должно быть взаимно простым с φ. Мы выбираем e=5 в качестве открытой экспоненты. Теперь у нас есть открытый ключ, который представляет собой пару чисел {e, n}. Эти числа мы предоставляем другим для шифрования сообщений. Однако, для нас это еще не завершено. Мы также должны получить закрытый ключ. Для этого нам нужно найти число d, которое является мультипликативно обратным к e по модулю φ. Мы хотим, чтобы (d × e) mod φ = 1. Мы выбираем d=17 в данной иллюстрации. Здесь d=17 обратное к e=5 по модулю φ=12, так как (17 × 5) mod 12 = 85 mod 12 = 1. Пара {d, n} составляет наш секретный ключ.

*Шаг второй. Шифрование*

Теперь, когда у нас есть открытый ключ {e, n}, и мы хотим зашифровать сообщение, представленное числом P. Для данной иллюстрации, допустим, что P=19. Мы выполняем шифрование с помощью следующего алгоритма: Мы возводим сообщение P в степень e по модулю n, что означает вычисление P в степени 5 (пропустим длинное значение и возьмем только остаток от деления на 21). В результате получаем зашифрованные данные E=10. Строго говоря, для вычислений мы используем только остаток от деления на n на каждом этапе умножения, чтобы избежать больших промежуточных значений. Но это детали реализации вычислений. Полученное значение E (10) представляет собой зашифрованные данные, которые мы отправляем.

*Шаг третий. Расшифровка*

Теперь, когда мы получили зашифрованные данные E (10), и у нас есть закрытый ключ {d, n}, мы можем приступить к расшифровке. Для расшифровки мы используем закрытый ключ, в частности, значение d. Мы возводим E в степень d и вычисляем остаток от деления на n, чтобы получить исходное сообщение P=19. Таким образом, расшифровка выполняется путем вычисления E в степени 17 (пропустим длинное значение и возьмем только остаток от деления на 21). Результатом является исходное сообщение P=19. Следует отметить, что только обладатель закрытого ключа может расшифровать сообщение, так как другие не имеют доступа к закрытому ключу. Это иллюстрирует принцип асимметричного шифрования.



1. В начале кода подключаем библиотеку:
   * System.Security.Cryptography - для работы с криптографическими операциями, в данном случае, с RSA.
2. Метод Encrypt:
   * Получает сообщение в виде строки и открытый ключ RSA в виде RSAParameters.
   * Создает новый объект RSACryptoServiceProvider и импортирует в него переданный открытый ключ.
   * Конвертирует текст сообщения в байты, используя кодировку UTF-8.
   * Затем шифрует сообщение с помощью открытого ключа и возвращает зашифрованные данные в виде массива байт.
3. Метод Decrypt:
   * Получает зашифрованные данные в виде массива байт и закрытый ключ RSA в виде RSAParameters.
   * Создает новый объект RSACryptoServiceProvider и импортирует в него переданный закрытый ключ.
   * Дешифрует зашифрованные данные с использованием закрытого ключа и возвращает результат в виде строки после конвертации обратно в текст с использованием кодировки UTF-8.

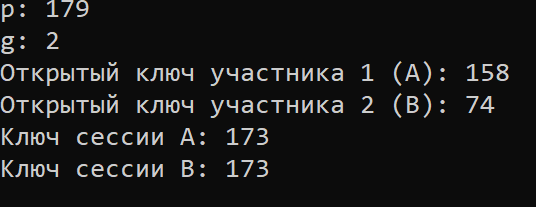
**2.Алгоритм Диффи-Хеллмана**

Обмен ключами с помощью протокола Диффи-Хеллмана (DH) – это метод безопасного обмена криптографическими ключами по общедоступному каналу. Это один из первых протоколов с открытым ключом, который изначально был концептуализирован Ральфом Мерклем и назван в честь Уитфилда Диффи и Мартина Хеллмана. DH является одним из первых практических примеров обмена открытыми ключами. Сегодня DH используется для многих приложений, таких как, например, Proton Mail, SSH, GPG и так далее.

Диффи-Хеллман работает по принципу неполного обмена ключом шифрования по сети. У каждой стороны есть открытый ключ (который может видеть каждый, включая предполагаемого хакера) и закрытый ключ (его может видеть только пользователь компьютера). У первого пользователя нет доступа к личному ключу второго. У второго – к первому. Положим, что оба используем общий алгоритм шифрования, и им нужно каким-то образом восстановить общий ключ шифрования, не передавая слишком много информации через сеть.

На практике обмен ключами по алгоритму Диффи‑Хеллмана происходит по следующей схеме.

1. Два участника обмена договариваются о двух числах. **Один выбирает большое простое число, а другой – целое число, меньшее числа первого участника**. Переговоры они могут вести открыто, и это никак не отразится на безопасности.
2. **Каждый из двух участников, независимо друг от друга, генерирует другое число, которое они будут хранить в тайне**. Эти числа выполняют роль секретного ключа. Далее в вычислениях используются секретный ключ и два предыдущих целых числа. Результат вычислений посылается участнику обмена, и он играет роль открытого ключа.
3. **Участники обмена обмениваются открытыми ключами.** Далее они, **используя собственный секретный ключ и открытый ключ партнера, конфиденциально вычисляют ключ сессии**. Каждый партер вычисляет один и тот же ключ сессии.
4. **Ключ сессии может использоваться как секретный ключ для другого алгоритма шифрования, например, DES**. Никакое третье лицо, контролирующее обмен, не сможет вычислить ключ сессии, не зная один из секретных ключей.

****

1. В классе DiffieHellman:
   * RunDiffieHellman():
     + Генерирует случайные простое число p и первообразный корень g.
     + Генерирует случайные закрытые ключи a и b для двух участников
     + Вычисляет открытые ключи A и B для каждого участника с использованием операции возведения в степень.
     + Участники обмениваются своими открытыми ключами.
     + Затем они вычисляют общий секретный ключ sharedKeyA и sharedKeyB с использованием своих закрытых ключей и чужих открытых ключей.
     + Выводит параметры и общие ключи на консоль.
2. Внутри класса DiffieHellman также есть вспомогательные методы:
   * GeneratePrimeNumber(): Генерирует случайное простое число p в заданном диапазоне.
   * IsPrime(int number): Проверяет, является ли число простым.
   * GeneratePrimitiveRoot(int p): Находит первообразный корень для заданного простого числа p.
3. В методе Main класса Program:
   * Запускается метод RunDiffieHellman() из класса DiffieHellman, который выполняет алгоритм Диффи-Хеллмана.

**3.Шифрование Эль-Гамаля**

Схема Эль-Гамаля – криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Фактически здесь используется схема Диффи-Хеллмана, чтобы сформировать общий секретный ключ для двух абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ. Для каждого следующего сообщения секретный ключ вычисляется заново.

## **Генерация ключей**

1. Генерируется случайное простое число ~p длины ~n [битов](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82).
2. Выбирается случайный примитивный элемент ~g.
3. Выбирается случайное целое число ~x такое, что ~1 < x < p-1.
4. Вычисляется ~y = g^x\,\bmod\,p.
5. Открытым ключом является тройка \left( p,g,y \right), закрытым ключом — число ~x.

**Шифрование**

1. Сообщение ~M шифруется следующим образом:
2. Выбирается сессионный [ключ](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D1%8E%D1%87_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)) — случайное целое число ~k такое, что ~1 < k < p - 1
3. Вычисляются числа a = g^k\,\bmod\,p и b = y^k M\,\bmod\,p.
4. Пара чисел \left( a, b \right) является [шифротекстом](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82" \o "Шифротекст).
5. Нетрудно видеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

**Расшифрование**

1. Зная закрытый ключ ~x, исходное сообщение можно вычислить из шифротекста \left( a, b \right) по формуле:

M = b(a^x)^{-1}\,\bmod\,p.

1. При этом нетрудно проверить, что

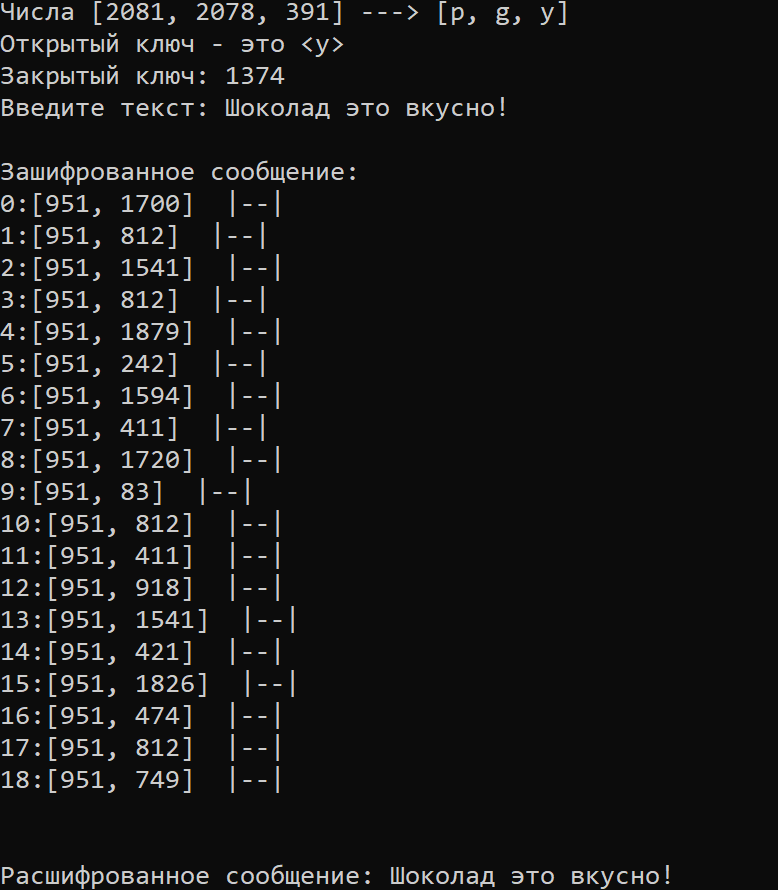
~(a^x)^{-1}\equiv g^{-kx}\pmod{p}

1. и поэтому

~b(a^x)^{-1}\equiv (y^kM)g^{-xk}\equiv (g^{xk}M) g^{-xk}\equiv M \pmod{p}.

1. Для практических вычислений больше подходит следующая формула:

M = b(a^x)^{-1}\,\bmod\,p = b \cdot a^{(p-1-x)}\,\bmod\,p 



1. Глобальные переменные:
   * g\_main: Главный генератор g для группы.
   * a: Секретное число a.
2. Метод Search\_g:
   * Этот метод используется для определения числа g, которое является генератором группы. Группа здесь используется в контексте алгоритма Эль-Гамаля.
   * Метод начинает с проверки, что число g возводимое в степени от 1 до p - 1 (где p - простое число) даёт разные остатки по модулю p.
   * Если это условие не выполняется, g уменьшается, и процесс начинается снова, пока не будет найдено подходящее g.
   * Результат сохраняется в глобальной переменной g\_main.
3. Метод Search\_p:
   * Этот метод генерирует простое число p.
   * Сначала он выбирает случайное число p из диапазона от 2000 до 2500.
   * Затем проверяет, что число p действительно является простым и что g\_main (генератор) можно найти для этого p.
   * Если условия не выполняются, выбирается новое случайное p.
4. Метод Cipher:
   * Этот метод используется для шифрования текста.
   * Он принимает текст для шифрования, простое число p и число y в качестве параметров.
   * Генерируется случайное число k из диапазона от 1 до p - 1.
   * Затем для каждого символа в тексте вычисляется a (с использованием g\_main и k) и шифруется символ с использованием y, k, и значения символа.
   * Зашифрованные значения сохраняются в списке array.
5. Метод Cipher\_RAZ:
   * Этот метод используется для расшифровки зашифрованного текста.
   * Принимает длину текста, список зашифрованных чисел, закрытый ключ x, и простое число p.
   * Для каждого зашифрованного числа выполняется операция расшифровки с использованием a, x, и p. Результат конвертируется в символ и добавляется к строке save\_text.

**Вывод**

Асимметричное шифрование – это метод шифрования данных, предполагающий использование двух ключей – открытого и закрытого. Открытый (публичный) ключ применяется для шифрования информации и может передаваться по незащищенным каналам. Закрытый (приватный) ключ применяется для расшифровки данных, зашифрованных открытым ключом. Открытый и закрытый ключи – это очень большие числа, связанные друг с другом определенной функцией, но так, что, зная одно, крайне сложно вычислить второе.

Асимметричное шифрование используется для защиты информации при ее передаче, также на его принципах построена работа электронных подписей.

Схема передачи данных между двумя субъектами (А и Б) с использованием открытого ключа выглядит следующим образом:

* Субъект А генерирует пару ключей, открытый и закрытый (публичный и приватный).
* Субъект А передает открытый ключ субъекту Б. Передача может осуществляться по незащищенным каналам.
* Субъект Б шифрует пакет данных при помощи полученного открытого ключа и передает его А. Передача может осуществляться по незащищенным каналам.
* Субъект А расшифровывает полученную от Б информацию при помощи секретного, закрытого ключа.

В такой схеме перехват любых данных, передаваемых по незащищенным каналам, не имеет смысла, поскольку восстановить исходную информацию возможно только при помощи закрытого ключа, известного лишь получателю и не требующего передачи.