Учреждение образования

«Белорусский государственный технологический университет»

**Основы информационной безопасности**

**Лабораторная работа №6**

**«Криптографическая защита информации**

**с помощью алгоритмов асимметричного**

**шифрования»**

Выполнил:

Студент 2 курса 2 группы ФИТ

Мойсеёнок Денис

Преподаватель:

Ржеутская Надежда Викентьевна

Минск 2025

**Цель**: овладеть основными криптографическими алгоритмами асимметричного шифрования.

**Вопросы:**

1. В чем особенность асимметричных криптосистем?

Ассиметричная система шифрования использует открытый ключ, который передается по открытому каналу и предназначен для шифрования сообщения. Шифрование на открытом ключе базируется на двух различных ключах, составляющих пару, но не идентичных.

2. С помощью какого ключа происходит шифрование сообщения?

Открытые ключи используются только для зашифровки данных.

3. В чем особенность алгоритма Диффи – Хеллмана?

Алгоритм Диффи – Хеллмана применяется для обработки небольших сообщений от отправителя получателю. Но в этом маленьком сообщении передается секретный ключ для расшифровки большого сообщения. Сильная сторона алгоритма – никто не сможет скомпрометировать секретное сообщение, зная один или даже два открытых ключа получателя и отправителя. В качестве секретных и открытых ключей используются очень большие целые числа

4. Какие недостатки можно выделить в асимметричных криптосистемах?

Недостатки в сравнении с симметричными системами:

− шифрование/расшифрование с использованием пары ключей проходит на два-три порядка медленнее, чем шифрование/расшифрование того же текста симметричным алгоритмом;

− требуются существенно бóльшие вычислительные ресурсы, поэтому на практике асимметричные криптосистемы используются в сочетании с другими алгоритмами;

− в алгоритм сложнее внести изменения;

− хотя сообщения надежно шифруются, получатель и отправитель самим фактом пересылки шифрованного сообщения «засвечиваются»;

− более длинные ключи.

**Задания:**

1. Объясните процесс работы алгоритма RSA

Генерация ключей:

* выбрать два простых различных числа p и q;
* вычислить модуль (произведение) n = pq;
* вычислить функцию Эйлера φ(n) = (p-1)(q-1);
* выбрать открытую экспоненту e;
* вычислить секретную экспоненту de = mod φ(n);
* опубликовать открытый ключ {e, n};
* сохранить закрытый ключ {d, n}.

Шифрование:

* выбрать текст для зашифровки m;
* вычислить шифротекст c = E(m) = m^(e) mod n.

Расшифрование:

* вычислить исходное сообщение m = D(c) = c^(d) mod n.

2. Объясните процесс работы алгоритма Диффи – Хеллмана.

* для начала два пользователя должны договориться о двух публичных числах p и g;
* каждый пользователь выбирает своё секретное число (закрытый ключ), которое никому не раскрывает. Первый пользователь число a, второй пользователь число b;
* теперь каждая сторона вычисляет свой открытый ключ на основе секретного числа и общих параметров. Первый пользователь A = g^(a) mod p, второй пользователь B = g^(b) mod p. Открытые ключи отправляются друг другу через открытый канал;
* теперь обе стороны используют полученный открытый ключ другой стороны и свой секретный ключ, чтобы вычислить общий секрет. Первый пользователь S = B^(a) mod p, второй пользователь S = A^(b) mod p.

3. Объясните процесс работы алгоритма Эль-Гамаля.

Генерация ключей:

* генерируется случайное простое число p длиной n битов;
* выбирается случайный примитивный элемент g;
* выбирается случайное целое число x такое, что 1 < x < p – 1;
* вычисляется число y = g^(x) mod p;
* открытым ключом является тройка (p, g, y), закрытым ключом – число x.

Шифрование:

* выбирается сессионный ключ – случайное целое число k такое, что

1 < k < p – 1;

* вычисляются числа a и b: a = g^(k) mod p, b = y^(k)M mod p;
* пара чисел (a, b) является шифротекстом.

Расшифрование:

* зная закрытый ключ x, исходное сообщение можно вычислить из шифротекста (a, b): M = b((a^(x))^(-1))mod p;

Для практических вычислений больше подходит формула:



4. Используя существующие криптографические библиотеки, создайте приложение и проанализировать работу вышеперечисленных алгоритмов.

Функция для алгоритма RSA:

|  |
| --- |
| def rsa\_example():  # Генерация ключей  private\_key = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048)  public\_key = private\_key.public\_key()  message = b"Secret message for RSA"  # Шифрование  ciphertext = public\_key.encrypt(  message,  padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  algorithm=hashes.SHA256(), label=None)  )  # Дешифрование  plaintext = private\_key.decrypt(  ciphertext,  padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  algorithm=hashes.SHA256(), label=None)  )  return plaintext == message |

Функция для алгоритма Диффи – Хеллмана:

|  |
| --- |
| def diffie\_hellman\_example():  # Параметры  parameters = dh.generate\_parameters(generator=2, key\_size=2048)  private\_key\_a = parameters.generate\_private\_key()  public\_key\_a = private\_key\_a.public\_key()  private\_key\_b = parameters.generate\_private\_key()  public\_key\_b = private\_key\_b.public\_key()  # Общий секрет для первого пользователя  shared\_key\_a = private\_key\_a.exchange(public\_key\_b)  # Общий секрет для второго пользователя  shared\_key\_b = private\_key\_b.exchange(public\_key\_a)  return shared\_key\_a == shared\_key\_b |

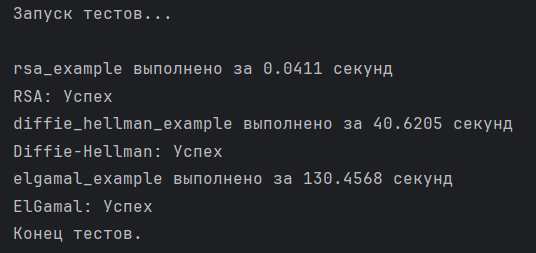
Класс для алгоритма Эль-Гамаля:

|  |
| --- |
| class ElGamal:  def \_\_init\_\_(self, key\_size=2048):  # Генерация чисел  parameters = dh.generate\_parameters(generator=2, key\_size=key\_size)  self.p = parameters.parameter\_numbers().p  self.g = parameters.parameter\_numbers().g  def generate\_keys(self):  # Генерация секретного ключа x и открытого ключа h  self.x = int.from\_bytes(os.urandom(32), 'big') % (self.p - 2) + 1 # Случайное x от 1 до p-2  self.h = pow(self.g, self.x, self.p)  return self.h  def encrypt(self, message):  m = int.from\_bytes(message.encode('utf-8'), 'big') % self.p  if m == 0:  m = 1  # Случайный ключ k  k = int.from\_bytes(os.urandom(32), 'big') % (self.p - 2) + 1  # Шифрование: c1, c2  c1 = pow(self.g, k, self.p)  c2 = (m \* pow(self.h, k, self.p)) % self.p  return (c1, c2)  def decrypt(self, ciphertext):  c1, c2 = ciphertext  # Дешифрование  s = pow(c1, self.x, self.p)  s\_inv = pow(s, self.p - 2, self.p)  m = (c2 \* s\_inv) % self.p  byte\_length = (m.bit\_length() + 7) // 8  m\_bytes = m.to\_bytes(byte\_length, 'big')  try:  return m\_bytes.decode('utf-8')  except UnicodeDecodeError:  return m |

Функция для алгоритма Эль-Гамаля:

|  |
| --- |
| def elgamal\_example():  # Инициализация  elgamal = ElGamal(key\_size=2048)  public\_key = elgamal.generate\_keys()  # Сообщение  message = "Secret ElGamal message"  # Шифрование  ciphertext = elgamal.encrypt(message)  # Дешифрование  decrypted = elgamal.decrypt(ciphertext)  return message == decrypted |

Результаты:



Выводы:

* RSA: Лучше всего подходит для шифрования небольших данных (например, ключей) и цифровых подписей. Быстрее в операциях шифрования, но медленнее в генерации ключей.
* Diffie-Hellman: Идеален для создания сессионных ключей в протоколах вроде TLS. Обеспечивает безопасность даже при компрометации долгосрочных ключей.
* ElGamal: Хорош для теоретического анализа и случаев, где требуется семантическая безопасность, но менее практичен из-за производительности и размера шифротекста.