Ковкель Никита, ФИТ 3-4

Информационная безопасность

Отчет по лабораторной работе № .10

Цель: изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров RSA и Эль-Гамаля.

**Практическая часть**

**1.** С помощью простого консольного приложения составить табличную или графическую форму зависимости времени вычисления параметра у, функционально заданного выражением вида:



|  |
| --- |
| def generate\_graphs(self):  message\_lengths = [13, 25, 50, 100, 300, 500, 1000]  rsa\_encrypt\_times = []  rsa\_encrypted\_lengths = []  elgamal\_encrypt\_times = []  elgamal\_encrypted\_lengths = []  for length in message\_lengths:  random\_text = ''.join(random.choices(string.ascii\_uppercase + string.digits, k=length))  def split\_message(message, chunk\_size):  return [message[i:i + chunk\_size] for i in range(0, len(message), chunk\_size)]  start\_time = time.time()  cipher = PKCS1\_OAEP.new(self.rsa\_key.publickey())  chunk\_size = int((self.rsa\_key.size\_in\_bytes() - 11) / 8)  rsa\_ciphertexts = []  for chunk in split\_message(random\_text.encode(), chunk\_size):  rsa\_ciphertexts.append(cipher.encrypt(chunk))  rsa\_encrypt\_times.append(time.time() - start\_time)  rsa\_encrypted\_lengths.append(len(rsa\_ciphertexts))  start\_time = time.time()  elgamal\_ciphertext, p = encryption(random\_text, self.q, self.h, self.g)  elgamal\_encrypt\_times.append(time.time() - start\_time)  elgamal\_encrypted\_lengths.append(len(str((elgamal\_ciphertext, p))))  # График времени шифрования  plt.figure(figsize=(10, 5))  plt.plot(message\_lengths, rsa\_encrypt\_times, label='RSA', color='green')  plt.plot(message\_lengths, elgamal\_encrypt\_times, label='Эль-Гамаль', color='orange')  plt.title('Зависимость y от длины сообщения')  plt.xlabel('Длина сообщения')  plt.ylabel('Время шифрования')  plt.legend()  plt.show() |

Листинг 10.1 – Код первого задания

2. Разработать авторское оконное приложение в соответствии с целью лабораторной работы.

1) RSA

Этот код содержит две функции, связанные с шифрованием и расшифровкой текста с помощью алгоритма RSA.

Функция handle\_rsa\_decrypt осуществляет расшифровку зашифрованного текста. Она сначала проверяет, используется ли base64 для представления зашифрованного текста, и преобразует его в байтовый формат соответствующим образом. Затем она использует RSA-ключ для расшифровки текста и выводит результат в текстовое поле. Также она отображает время расшифровки, длину исходного и зашифрованного текста.

Функция handle\_rsa реализует шифрование текста с помощью RSA. Она берет открытый текст, шифрует его с использованием открытого ключа RSA и, при необходимости, кодирует зашифрованный текст в base64. Результат выводится в текстовое поле, и отображается время шифрования, длина исходного и зашифрованного текста.

|  |
| --- |
| def handle\_rsa\_decrypt(self, ciphertext, use\_base64):  try:  start\_time = time.time()  if use\_base64:  ciphertext = decode\_base64(ciphertext)  else:  ciphertext = bytes.fromhex(ciphertext)  cipher = PKCS1\_OAEP.new(self.rsa\_key)  decrypted\_text = cipher.decrypt(ciphertext)  self.text\_area.delete("1.0", "end")  self.text\_area.insert("1.0", decrypted\_text.decode())  elapsed\_time = time.time() - start\_time  original\_length = len(decrypted\_text)  encrypted\_length = len(ciphertext)  messagebox.showinfo("Результаты",  f"Время расшифр.: {elapsed\_time:.6f} секунд\nДлина ориг. текста: {original\_length} байт\nДлина зашифр. текста: {encrypted\_length} байт")  except Exception as e:  messagebox.showerror("Ошибка", str(e))  def handle\_rsa(self, plaintext, use\_base64):  try:  start\_time = time.time()  cipher = PKCS1\_OAEP.new(self.rsa\_key.publickey())  ciphertext = cipher.encrypt(plaintext.encode())  if use\_base64: # Проверяем, используется ли base64  ciphertext = encode\_base64(ciphertext) # Кодируем шифртекст в base64  else:  ciphertext = ciphertext.hex()  self.text\_area.delete("1.0", "end")  self.text\_area.insert("1.0", ciphertext)  elapsed\_time = time.time() - start\_time  original\_length = len(plaintext)  encrypted\_length = len(ciphertext)  messagebox.showinfo("Результаты",  f"Время зашифр.: {elapsed\_time:.6f} секунд\nДлина ориг. текста: {original\_length} байт\nДлина зашифр. текста: {encrypted\_length} байт")  except Exception as e:  messagebox.showerror("Ошибка", str(e)) |

Листинг 10.2 – Код для зашифрования и расшифрования RSA



Рисунок 10.1 - Результат выполнения зашифрования и расшифрования на основе алгоритма RSA

2) Эль-Гамаль

Данный код содержит две функции, связанные с шифрованием и расшифровкой текста с помощью алгоритма ElGamal.

Функция handle\_elgamal реализует шифрование текста с использованием алгоритма ElGamal. Она берет открытый текст, вызывает функцию encryption для получения зашифрованного текста и параметра p, и затем записывает эти данные в текстовое поле. Также она отображает время шифрования, длину исходного и зашифрованного текста.

Функция handle\_elgamal\_decrypt осуществляет расшифровку зашифрованного текста. Она проверяет, используется ли base64 для представления зашифрованного текста, и преобразует данные в соответствующий формат. Затем она вызывает функцию decryption для расшифровки текста с использованием закрытого ключа и записывает результат в текстовое поле. Также она отображает время расшифровки, длину исходного и зашифрованного текста.

|  |
| --- |
| def handle\_elgamal(self, plaintext, use\_base64):  try:  start\_time = time.time()  ciphertext, p = encryption(plaintext, self.q, self.h, self.g)  if use\_base64:  ciphertext\_str = encode\_base64(str(ciphertext))  p\_str = encode\_base64(str(p))  ciphertext\_data = (ciphertext\_str, p\_str)  else:  ciphertext\_data = (ciphertext, p)  self.text\_area.delete("1.0", "end")  self.text\_area.insert("1.0", str(ciphertext\_data))  elapsed\_time = time.time() - start\_time  original\_length = len(plaintext)  encrypted\_length = len(str(ciphertext\_data))  messagebox.showinfo("Результаты",  f"Время зашифр.: {elapsed\_time:.6f} секунд\nДлина ориг. текста: {original\_length} байт\nДлина зашифр. текста: {encrypted\_length} байт")  except Exception as e:  messagebox.showerror("Ошибка", str(e))  def handle\_elgamal\_decrypt(self, ciphertext, use\_base64):  try:  start\_time = time.time()  if use\_base64:  ciphertext\_data = eval(ciphertext)  ciphertext\_str, p\_str = ciphertext\_data  ciphertext = eval(decode\_base64(ciphertext\_str))  p = eval(decode\_base64(p\_str))  else:  ciphertext\_data = eval(ciphertext)  ciphertext, p = ciphertext\_data  plaintext = decryption(ciphertext, p, self.key, self.q)  plaintext\_str = ''.join(plaintext)  self.text\_area.delete("1.0", "end")  self.text\_area.insert("1.0", plaintext\_str)  elapsed\_time = time.time() - start\_time  original\_length = len(plaintext\_str)  encrypted\_length = len(ciphertext)  messagebox.showinfo("Результаты",  f"Время расшифр.: {elapsed\_time:.6f} секунд\nДлина ориг. текста: {original\_length} байт\nДлина зашифр. текста: {encrypted\_length} байт")  except Exception as e:  messagebox.showerror("Ошибка", str(e)) |

Листинг 10.3 – Зашифрование и расшифрование алгоритм Эль-Гамаля

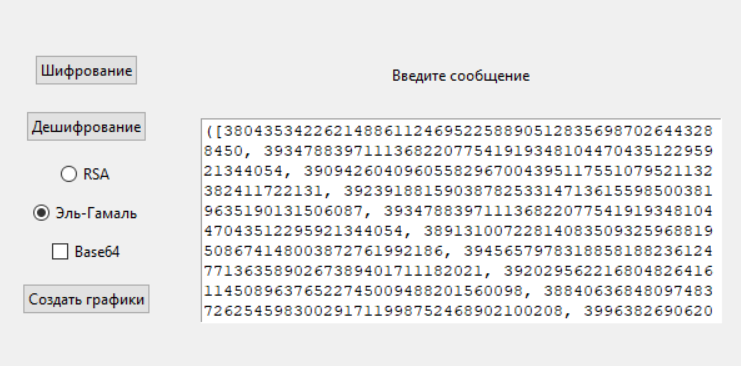


Рисунок 10.2 - Результат выполнения зашифрования и расшифрования на основе алгоритма Эль-Гаммаля

3. Время выполнения

Для RSA шифрования, сначала отмечается время начала операции. Для шифрования с помощью алгоритма ElGamal, сначала отмечается время начала операции. А после выполнения операции время останавливается и выводится результат на экран.

|  |
| --- |
| start\_time = time.time()  cipher = PKCS1\_OAEP.new(self.rsa\_key.publickey())  chunk\_size = int((self.rsa\_key.size\_in\_bytes() - 11) / 8)  rsa\_ciphertexts = []  for chunk in split\_message(random\_text.encode(), chunk\_size):  rsa\_ciphertexts.append(cipher.encrypt(chunk))  rsa\_encrypt\_times.append(time.time() - start\_time)  rsa\_encrypted\_lengths.append(len(rsa\_ciphertexts))  start\_time = time.time()  elgamal\_ciphertext, p = encryption(random\_text, self.q, self.h, self.g)  elgamal\_encrypt\_times.append(time.time() - start\_time)  elgamal\_encrypted\_lengths.append(len(str((elgamal\_ciphertext, p)))) |

Листинг 10.3 – Зашифрование и расшифрование алгоритм Эль-Гамаля

**Вывод:** в результате работы были закреплены теоретические знания по алгоритмам RSA и Эль-Гамаля, разработано приложение для асимметричного шифрования, проведен анализ криптостойкости алгоритмов и оценена их производительность. Полученные выводы подтверждают важность изучения и применения асимметричного шифрования для обеспечения безопасности данных.