Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 4

**Компьютерная реализация алгоритма шифрования Эль-Гамаля.**

Выполнил студент гр. 953501

Трофимук Г.А

Проверил

Протько М.И.

Минск, 2022

# 1. Введение

Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма Эль-Гамаля.

**2. Теоретические сведения**

**Алгоритм Эль-Гамаля**

Схема Эль-Гамаля (Elgamal) — криптосистема с открытым ключом, основанная на трудности вычисления дискретных логарифмов в конечном поле. Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов электронной цифровой подписи в США (DSA) и России (ГОСТ Р 34.10-94).

**Генерация ключей:**

* Генерируется случайное простое число p.
* Выбирается целое число g — первообразный корень p.
* Выбирается случайное целое число x такое, что 1 < x < p - 1.
* Вычисляется y = g^x mod p.
* Открытым ключом является y, закрытым ключом — число x.

**Шифрование**

Сообщение M должно быть меньше числа p. Сообщение шифруется следующим образом:

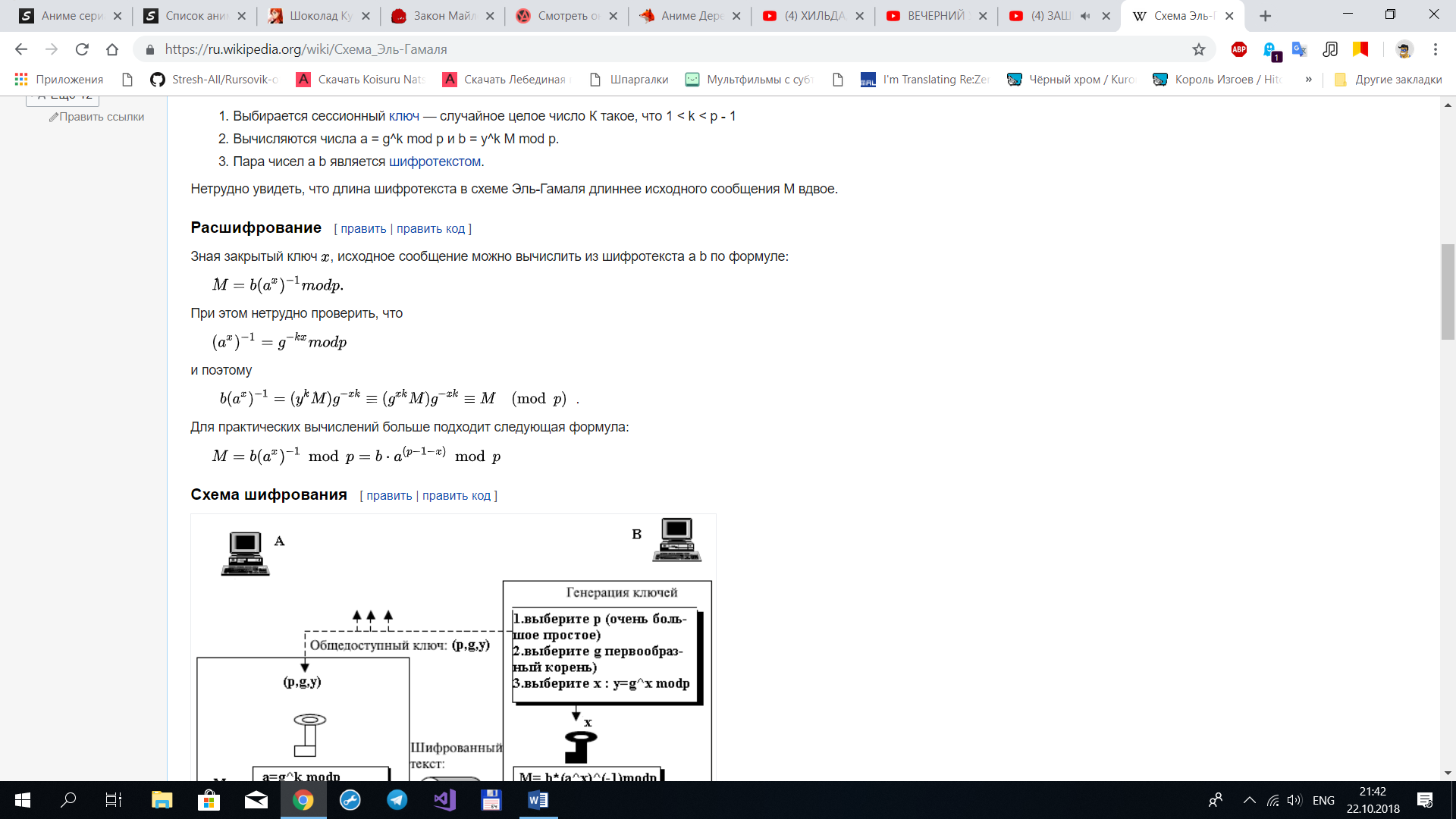
1. Выбирается сессионный ключ — случайное целое число К такое, что

1 < k < p - 1

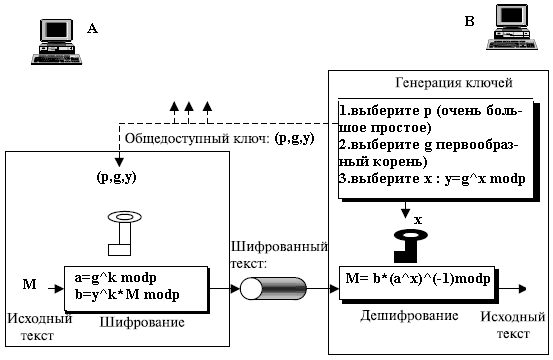
1. Вычисляются числа a = g^k mod p и b = y^k M mod p.
2. Пара чисел a b является шифротекстом.

Нетрудно увидеть, что длина шифротекста в схеме Эль-Гамаля длиннее исходного сообщения M вдвое.

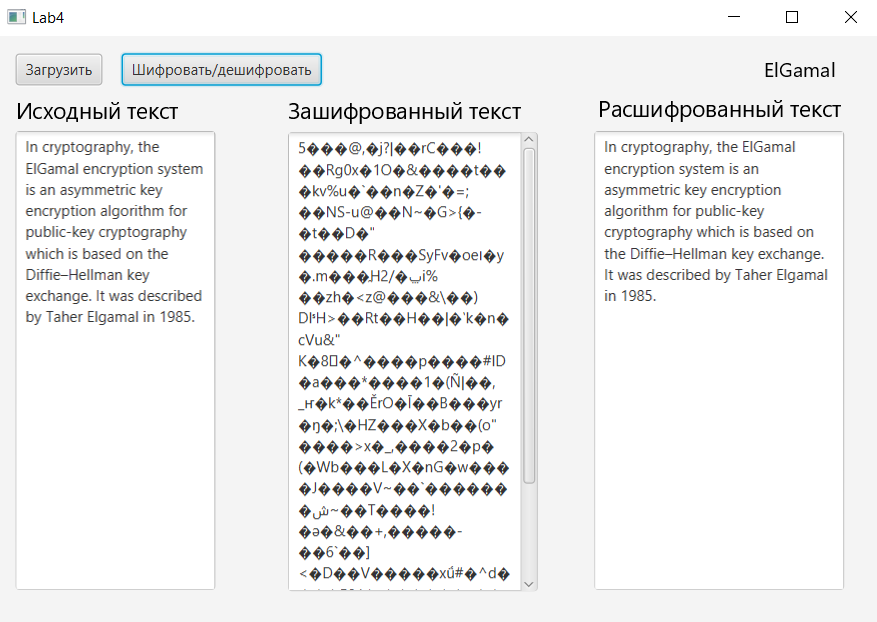
**Расшифрование**

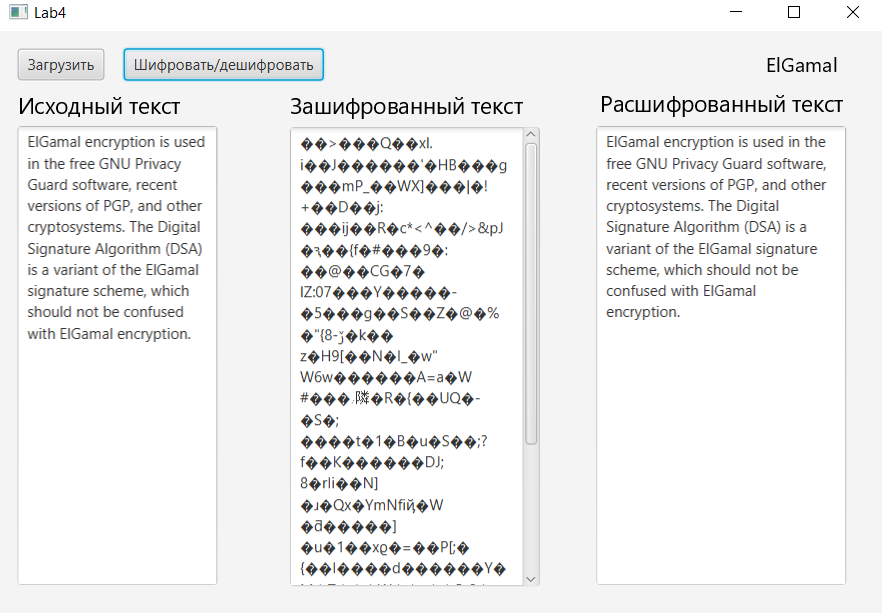


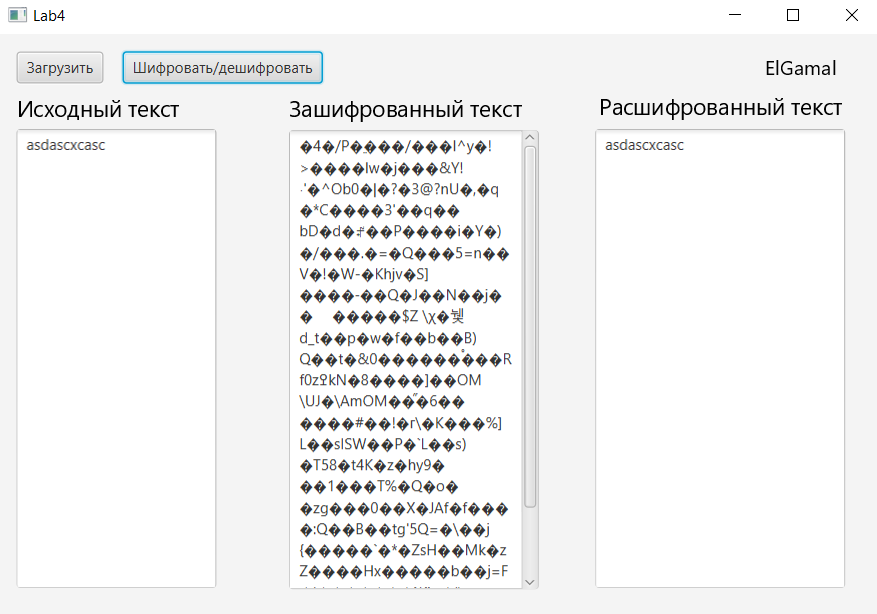
**Схема шифрования**

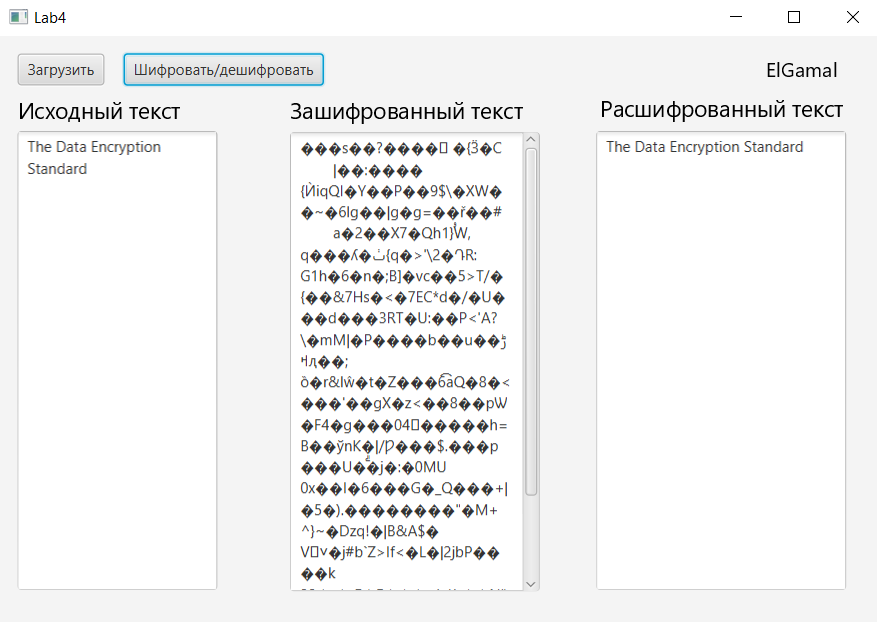


# 3. Результаты выполнения программы









# Выводы

Схема Эль-Гамаля (Elgamal) — [криптосистема](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) с открытым ключом, основанная на трудности вычисления [дискретных логарифмов](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC) в [конечном поле](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5). Криптосистема включает в себя алгоритм шифрования и алгоритм цифровой подписи. Схема Эль-Гамаля лежит в основе бывших стандартов [электронной цифровой подписи](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) в [США](https://ru.wikipedia.org/wiki/%D0%A1%D0%A8%D0%90) ([DSA](https://ru.wikipedia.org/wiki/DSA)) и [России](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F) ([ГОСТ Р 34.10-94](https://ru.wikipedia.org/wiki/%D0%93%D0%9E%D0%A1%D0%A2_%D0%A0_34.10-94)).

Схема была предложена [Тахером Эль-Гамалем](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D1%85%D0%B5%D1%80_%D0%AD%D0%BB%D1%8C-%D0%93%D0%B0%D0%BC%D0%B0%D0%BB) в [1985 году](https://ru.wikipedia.org/wiki/1984_%D0%B3%D0%BE%D0%B4). Эль-Гамаль разработал один из вариантов [алгоритма Диффи-Хеллмана](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B8%D1%84%D1%84%D0%B8_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0). Он усовершенствовал систему Диффи-Хеллмана и получил два алгоритма, которые использовались для шифрования и для обеспечения аутентификации. В отличие от RSA, алгоритм Эль-Гамаля не был запатентован и поэтому стал более дешёвой альтернативой, так как не требовалась оплата взносов за лицензию. Считается, что алгоритм попадает под действие патента Диффи-Хеллмана.

# КОД ПРОГРАММЫ

**Main.java:**

package com.example.lab4si;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
import org.bouncycastle.jce.provider.BouncyCastleProvider;  
  
import java.io.IOException;  
import java.security.Security;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 stage.setTitle("Lab4");  
 stage.setScene(scene);  
 stage.show();  
 Controller controller = fxmlLoader.getController();  
 controller.setStage(stage);  
 Security.*addProvider*(new BouncyCastleProvider());  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

**Controller.java:**

package com.example.lab4si;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.TextArea;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
  
import javax.crypto.BadPaddingException;  
import javax.crypto.Cipher;  
import javax.crypto.IllegalBlockSizeException;  
import javax.crypto.NoSuchPaddingException;  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
import java.security.\*;  
  
public class Controller {  
 private Stage stage;  
 @FXML  
 private TextArea sourceTextArea;  
 @FXML  
 private TextArea encryptElGamalTextArea;  
 @FXML  
 private TextArea decryptElGamalTextArea;  
  
 public Stage getStage() {  
 return stage;  
 }  
  
 public void setStage(Stage stage) {  
 this.stage = stage;  
 }  
 @FXML  
 void Load(ActionEvent event) throws IOException {  
 final FileChooser fileChooser = new FileChooser();  
 fileChooser.setInitialDirectory(new File("C:\\Users\\User\\Desktop\\Tests\\Txts"));  
 File file = fileChooser.showOpenDialog(stage);  
 if (file != null) {  
 FileReader reader = new FileReader(file.getAbsolutePath());  
 StringBuilder result = new StringBuilder();  
 int character;  
 while((character=reader.read())!=-1) {  
 result.append((char) character);  
 }  
 sourceTextArea.setText(result.toString());  
 }  
 }  
 @FXML  
 void EncryptDecrypt(ActionEvent event) throws NoSuchAlgorithmException, NoSuchProviderException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException {  
 String message = sourceTextArea.getText();  
 KeyPairGenerator keyGen = KeyPairGenerator.*getInstance*("ElGamal", "BC");  
 SecureRandom random = new SecureRandom();  
 keyGen.initialize(2048, random);  
 Cipher cipher = Cipher.*getInstance*("ElGamal/None/NoPadding", "BC");  
 KeyPair keys = keyGen.generateKeyPair();  
 Key pubKey = keys.getPublic();  
 Key privKey = keys.getPrivate();  
 cipher.init(Cipher.*ENCRYPT\_MODE*, pubKey, random);  
 byte[] cipherText = cipher.doFinal(message.getBytes());  
 encryptElGamalTextArea.setText(new String(cipherText));  
 cipher.init(Cipher.*DECRYPT\_MODE*, privKey);  
 byte[] resultText = cipher.doFinal(cipherText);  
 decryptElGamalTextArea.setText(new String(resultText));  
 }  
}