Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 7

**Криптография с использованием эллиптических кривых**

Выполнил студент гр. 953501

Трофимук Г.А

Проверил

Протько М.И.

Минск, 2022

# 1. Введение

Реализовать программное средство формирования электронной цифровой подписи на основе эллиптических кривых ECDSA и программное средство и программное средство, реализующее простой подход к шифрованию/дешифрованию с использованием *эллиптических кривых*.

**2. Теоретические сведения**

**Математические понятия**

Преимущество подхода на основе *эллиптических кривых* в сравнении с задачей факторизации числа, используемой в RSA, или задачей целочисленного логарифмирования, применяемой в алгоритме Диффи-Хеллмана и в DSS, заключается в том, что в данном случае обеспечивается эквивалентная защита при меньшей длине ключа.

В общем случае уравнение *эллиптической кривой*   Е имеет вид:

y2 + axy + by = x3 + cx2 + dx + e

В качестве примера рассмотрим *эллиптическую кривую*   Е, уравнение которой имеет вид: y2 + y = x3 - x2

На этой кривой лежат только четыре точки, координаты которых являются целыми числами. Это точки

А (0, 0), В (1, -1), С (1, 0) и D (0, -1)

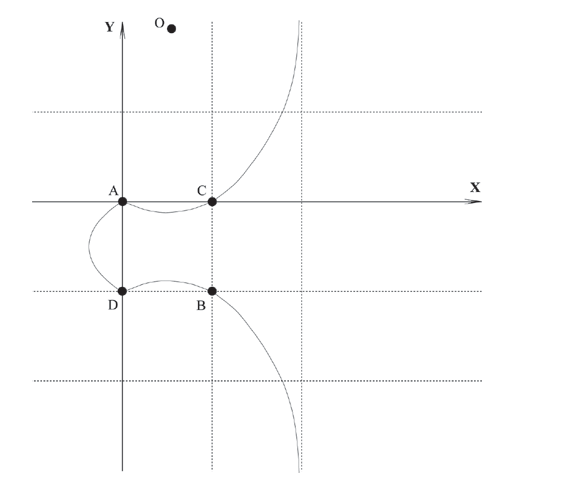


Рисунок 1 – Пример эллиптической кривой с четырьмя точками

Для определения *операции сложения для точек на эллиптической кривой* сделаем следующие предположения:

                На плоскости существует бесконечно удаленная точка 0Е, в которой сходятся все вертикальные прямые.

                Будем считать, что касательная к кривой пересекает точку касания два раза.

                Если три точки *эллиптической кривой* лежат на прямой линии, то их сумма есть 0.

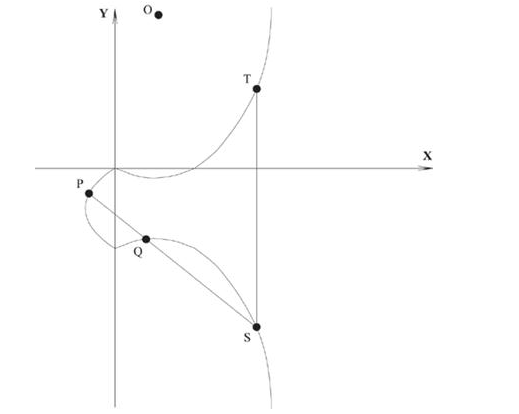


Рисунок 2 – Сложение точек на эллиптической кривой

Введем следующие правила сложения точек на *эллиптической кривой*:

                Точка 0 выступает в роли *нулевого элемента*. Так, 0 = -0 и для любой точки Р на *эллиптической кривой* Р + 0 = Р.

                Вертикальная линия пересекает кривую в двух точках с одной и той же координатой х - скажем, S = (x, y) и T = (x, -y). Эта прямая пересекает кривую и в бесконечно удаленной точке. Поэтому Р1 + Р2 + 0 = 0 и Р1 = -Р2.

                Чтобы сложить две точки P и Q (см. рисунок 11.2) с разными координатами х, следует провести через эти точки прямую и найти точку пересечения ее с *эллиптической кривой*. Если прямая не является касательной к кривой в точках P или Q, то существует только одна такая точка, обозначим ее S. Согласно нашему предположению P + Q + S = О

Следовательно, P + Q = -S или P + Q = T.

Если прямая является касательной к кривой в какой-либо из точек P или Q, то в этом случае следует положить S = P или S = Q соответственно.

                Чтобы удвоить точку Q, следует провести касательную в точке Q и найти другую точку пересечения S с *эллиптической кривой*. Тогда Q + Q = 2 × Q = -S.

Введенная таким образом *операция сложения* подчиняется всем обычным правилам сложения, в частности коммутативному и ассоциативному законам. Умножение точки Р *эллиптической кривой* на положительное число k определяется как сумма k точек Р.

В криптографии с использованием *эллиптических кривых* все значения вычисляются по модулю р, где р является простым числом. Элементами данной *эллиптической кривой* являются пары неотрицательных целых чисел, которые меньше р и удовлетворяют частному виду *эллиптической кривой*:

y2 ≡x3 + ax + b (mod p)

Такую кривую будем обозначать Ep (a,b). При этом числа а и b должны быть меньше р и должны удовлетворять условию 4a3 + 27b2 (mod p) ≠ 0. Множество точек на *эллиптической кривой* вычисляется следующим образом.

Для каждого такого значения х, что 0≤х≤р, вычисляется x3 + ax + b (mod p).

Для каждого из полученных на предыдущем шаге значений выясняется, имеет ли это значение квадратный корень по модулю р. Если нет, то в Ep (a,b) нет точек с этим значением х. Если корень существует, имеется два значения y, соответствующих операции извлечения квадратного корня (исключением является случай, когда единственным значением оказывается y = 0). Эти значения (x,y) и будут точками Ep (a,b).

Множество точек Ep (a,b) обладает следующими свойствами:

1.           Р + 0 = Р

2.           Если Р = (x,y), то Р + (x,-y) = 0. Точка (x,-y) является отрицательным значением точки Р и обозначается -Р. Заметим, что (x,-y) лежит на *эллиптической кривой* и принадлежит Ep (a,b).

3.           Если Р = (x1,y1) и Q = (x2,y2), где P ≠ Q, то P + Q = (x3,y3) определяется по следующим формулам:

4.           x3≡  λ2 - x1 - x2 (mod p)

5.           y3 ≡ λ (x1 - x3) - y1 (mod p)

где

      (y2 - y1)/(x2 - x1) , если P ≠ Q

 λ =       (3x12 + a)/2y1 , если P = Q

Число λ есть угловой коэффициент секущей, проведенной через точки P = (x1, y1) и Q = (x2, y2). При P = Q секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

Задача, которую должен решить в этом случае атакующий, есть своего рода задача ***"дискретного логарифмирования на эллиптической кривой"***, и формулируется она следующим образом. Даны точки P и Q на *эллиптической кривой*   Ep (a,b). Необходимо найти коэффициент k < p такой, что

P = k × Q

Относительно легко вычислить P по данным k и Q, но довольно трудно вычислить k, зная P и Q.

**Аналог алгоритма Диффи-Хеллмана обмена ключами**

Обмен ключами с использованием *эллиптических кривых* может быть выполнен следующим образом. Сначала выбирается простое число р ≈ 2180 и параметры a и b для уравнения *эллиптической кривой*. Это задает множество точек Ep (a,b). Затем в Ep (a,b) выбирается генерирующая точка G = (x1,y1). При выборе G важно, чтобы наименьшее значение n, при котором n × G = 0, оказалось очень большим простым числом. Параметры Ep (a,b) и G криптосистемы являются параметрами, известными всем участникам.

Обмен ключами между пользователями А и В производится по следующей схеме.

1.           Участник А выбирает целое число nA, меньшее n. Это число является закрытым ключом участника А. Затем участник А вычисляет открытый ключ PA = nA × G, который представляет собой некоторую точку на Ep (a,b).

2.           Точно так же участник В выбирает закрытый ключ nB и вычисляет открытый ключ PB.

3.           Участники обмениваются открытыми ключами, после чего вычисляют общий секретный ключ K

Участник А: K = nA × PB

       Участник В: K = nВ × PА

Следует заметить, что общий секретный ключ представляет собой пару чисел. Если данный ключ предполагается использовать в качестве сеансового ключа для алгоритма симметричного шифрования, то из этой пары необходимо создать одно значение.

**Алгоритм цифровой подписи на основе эллиптических кривых ECDSA**

Алгоритм ECDSA (Elliptic Curve Digest Signature Algorithm) принят в качестве стандартов ANSI X9F1 и IEEE P1363.

Создание ключей:

1.           Выбирается *эллиптическая кривая*   Ep (a,b). Число точек на ней должно делиться на большое целое n.

2.           Выбирается точка РEp (a,b).

3.           Выбирается случайное число d  [1, n-1].

4.           Вычисляется Q = d × P.

5.           Закрытым ключом является d, открытым ключом – (E, P, n, Q).

Создание подписи:

1.           Выбирается случайное число k [1, n-1].

2.           Вычисляется k × P = (x1, y1)   и  r = x1 (mod n).

Проверяется, чтобы r не было равно нулю, так как в этом случае подпись не будет зависеть от закрытого ключа. Если r = 0, то выбирается другое случайное число k.

3.           Вычисляется k-1 mod n

4.           Вычисляется s = k-1 (Н(M) + dr) (mod n)

Проверяется, чтобы s не было равно нулю, так как в этом случае необходимого для проверки подписи числа s-1 mod n не существует. Если s = 0, то выбирается другое случайное число k.

Подписью для сообщения М является пара чисел (r,s).

Проверка подписи:

1.           Проверить, что целые числа r и s принадлежат диапазону чисел [0, n-1]. В противном случае результат проверки отрицательный, и подпись отвергается.

2.           Вычислить w = s-1 (mod n) и H(M)

3.           Вычислить u1 = H(M) w (mod n), u2 = rw (mod n)

4.           Вычислить  u1P + u2Q = (x0, y0), v = x0 (mod n)

5.           Подпись верна в том и только том случае, когда v = r.

**Шифрование/дешифрование с использованием эллиптических кривых**

Рассмотрим самый простой подход к шифрованию/дешифрованию с использованием *эллиптических кривых*. Задача состоит в том, чтобы зашифровать сообщение М, которое может быть представлено в виде точки на эллиптической кривой Pm (x,y).

Как и в случае обмена ключом, в системе шифрования/дешифрования в качестве параметров рассматривается *эллиптическая кривая*   Ep (a,b) и точка G на ней. Участник B выбирает закрытый ключ nB и вычисляет открытый ключ PB = nB × G. Чтобы зашифровать сообщение Pm используется открытый ключ получателя B   PB. Участник А выбирает случайное целое положительное число k и вычисляет зашифрованное сообщение Cm, являющееся точкой на *эллиптической кривой*.

Cm = {k × G, Pm + k × PB}

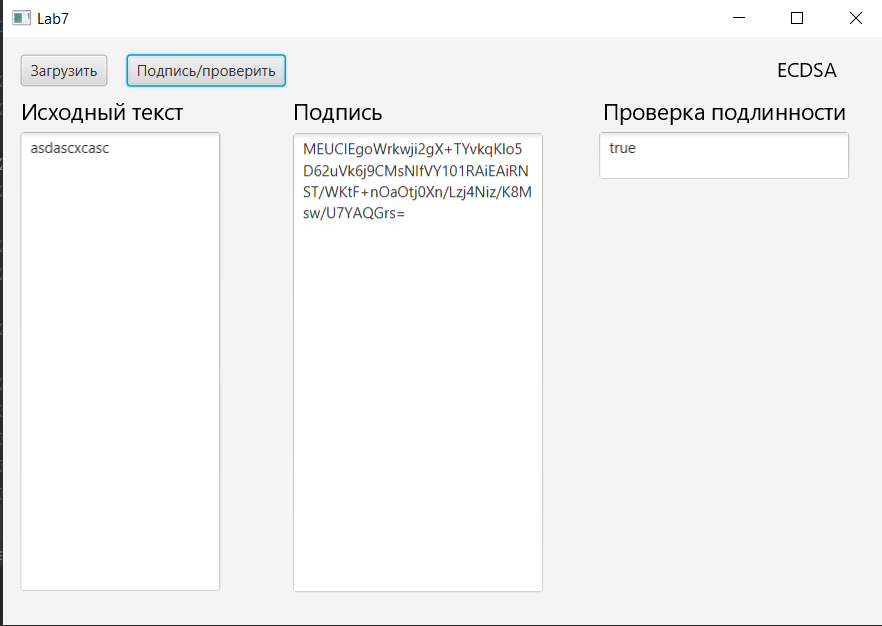
Чтобы дешифровать сообщение, участник В умножает первую координату точки на свой закрытый ключ и вычитает результат из второй координаты:

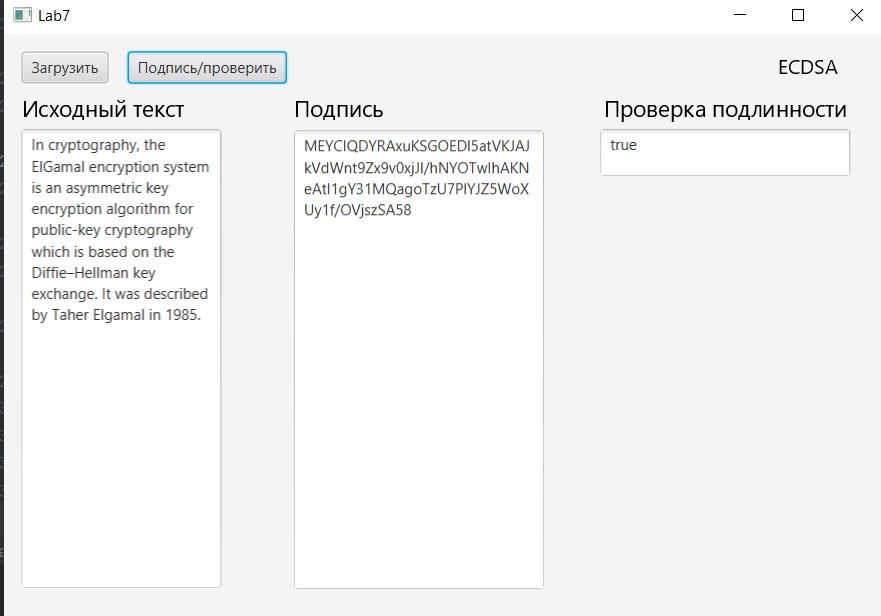
Pm + k × PB - nB × (k × G) = Pm + k × (nB × G) - nB × (k × G) = Pm

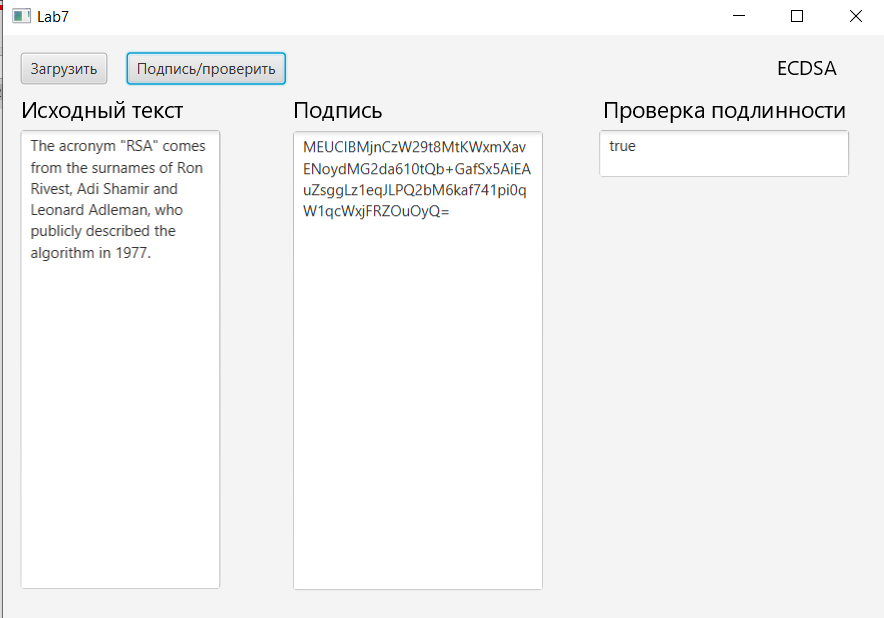
Участник А зашифровал сообщение Pm добавлением к нему kxPB. Никто не знает значения k, поэтому, хотя PB и является открытым ключом, никто не знает k × PB. Противнику для восстановления сообщения придется вычислить k, зная G и k × G. Сделать это будет нелегко.

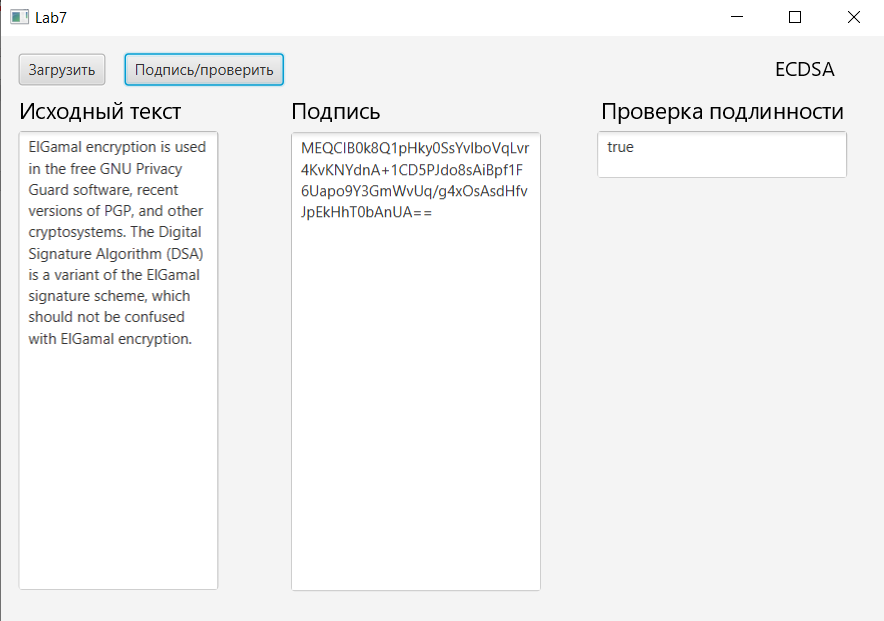
Получатель также не знает k, но ему в качестве подсказки посылается k × G. Умножив k × G на свой закрытый ключ, получатель получит значение, которое было добавлено отправителем к незашифрованному сообщению. Тем самым получатель, не зная k, но имея свой закрытый ключ, может восстановить незашифрованное сообщение.

# 3. Результаты выполнения программы









# Выводы

ЭЦП — это реквизит [электронного документа](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82), полученный в результате криптографического преобразования [информации](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) с использованием [закрытого ключа](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи (целостность), принадлежность подписи владельцу [сертификата ключа подписи](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D1%8E%D1%87%D0%B0) (авторство), а в случае успешной проверки подтвердить факт подписания электронного документа (неотказуемость).

ECDSA — алгоритм с [открытым ключом](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_%D1%81_%D0%BE%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%BC_%D0%BA%D0%BB%D1%8E%D1%87%D0%BE%D0%BC), использующийся для построения и проверки [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_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C) при помощи [криптографии на эллиптических кривых](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F).

ECDSA используется в криптовалютных транзакциях (например, в [биткойне](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82%D0%BA%D0%BE%D0%B9%D0%BD) и [эфириуме](https://ru.wikipedia.org/wiki/Ethereum)) для обеспечения того, чтобы средства могли быть потрачены только их законными владельцами.

Стойкость алгоритма шифрования основывается на [проблеме дискретного логарифма в группе точек эллиптической кривой](https://ru.wikipedia.org/wiki/ECDLP). В отличие от [проблемы простого дискретного логарифма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B5_%D0%BB%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [проблемы факторизации целого числа](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), не существует [субэкспоненциального алгоритма](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) для проблемы дискретного логарифма в группе точек эллиптической кривой. По этой причине «сила на один бит ключа» существенно выше в алгоритме, который использует эллиптические кривые.

Это означает, что в криптографии на эллиптических кривых можно использовать значительно меньшие параметры, чем в других системах с открытыми ключами, таких как [RSA](https://ru.wikipedia.org/wiki/RSA) и [DSA](https://ru.wikipedia.org/wiki/DSA), но с эквивалентным уровнем безопасности. К примеру, битовый размер [ключей](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)): 160-битный ключ будет равносилен ключам с 1024-битным модулем в RSA и DSA при сопоставимом уровне безопасности (против известных атак).

Преимущества, полученные от меньших размеров параметров (в частности, ключей), включают скорость выполнения алгоритма, эффективное использование энергии, пропускной полосы, памяти. Они особенно важны для приложений на устройствах с ограниченными возможностями, таких как [смарт-карты](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D1%80%D1%82-%D0%BA%D0%B0%D1%80%D1%82%D0%B0).

# КОД ПРОГРАММЫ

**Main.java:**

package com.example.lab7si;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
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("Lab7");  
 stage.setScene(scene);  
 stage.show();  
 Controller controller = fxmlLoader.getController();  
 controller.setStage(stage);  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

**Controller.java:**

package com.example.lab7si;  
  
import com.starkbank.ellipticcurve.Ecdsa;  
import com.starkbank.ellipticcurve.PrivateKey;  
import com.starkbank.ellipticcurve.PublicKey;  
import com.starkbank.ellipticcurve.Signature;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.TextArea;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
  
import java.io.File;  
import java.io.FileReader;  
import java.io.IOException;  
  
  
public class Controller {  
 private Stage stage;  
 @FXML  
 private TextArea sourceTextArea;  
 @FXML  
 private TextArea signedTextArea;  
 @FXML  
 private TextArea checkTextArea;  
 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 Exception {  
 // Generate Keys  
 PrivateKey privateKey = new PrivateKey();  
 PublicKey publicKey = privateKey.publicKey();  
  
 String message = sourceTextArea.getText();  
 // Generate Signature  
 Signature signature = Ecdsa.*sign*(message, privateKey);  
  
 signedTextArea.setText(signature.toBase64());  
  
 // Verify if signature is valid  
 boolean verified = Ecdsa.*verify*(message, signature, publicKey) ;  
  
 // Return the signature verification status  
 checkTextArea.setText(String.*valueOf*(verified));  
 }  
}