**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИБ**

отчет

**по научно-исследовательской работе**

Тема: **Разработка приложения, демонстрирующего работу**

**блочного шифра "Кузнечик"**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2363 |  | Катенко Ю.В. |
| Руководитель |  | Племянников А.К. |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на научно-исследовательскую работу**

|  |  |  |
| --- | --- | --- |
| Студентка Катенко Ю.В. | | |
| Группа 2363 | | |
| Тема НИР: Разработка приложения, демонстрирующего работу блочного шифра "Кузнечик" | | |
| Задание на НИР:  Изучить криптографический алгоритм «Кузнечик», разработать приложение, демонстрирующее его работу, внедрить это приложение в криптографическую платформу JCrypTool. | | |
|  | | |
| Дата сдачи отчета: 26.12.2016 | | |
| Дата защиты отчета: 26.12.2016 | | |
|  | | |
| Студентка |  | Катенко Ю.В. |
| Руководитель |  | Племянников А.К. |

**Аннотация**

Целью данной научно-исследовательской работы является создание приложения, наглядно демонстрирующего работу криптографического алгоритма «Кузнечик», описанного в ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры».

В работе приводится описание алгоритма демонстрационного приложения и графическая схема, отображающая все его этапы, а также изучаются возможности создания криптографического плагина и его внедрения в обучающую платформу JCrypTool.

**Summary**

Purpose of the research work is to create application for demonstration functioning of "Kuznyechik" algorithm which described in GOST R 34.12-2015 "Information technology. Cryptographical information protection. Block ciphers."

The research work contains description of algorithm of the demonstrational application and graphical schemes showing all of its steps. Also, possibilities of creating cryptographical plugin and its integration into Jcryptool platform was explored.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc470171202)

[1. ПОСТАНОВКА ЗАДАЧИ 6](#_Toc470171203)

[2. ЛОГИКА ПРИЛОЖЕНИЯ 7](#_Toc470171204)

[3. СОЗДАНИЕ ПЛАГИНА И ВНЕДРЕНИЕ ЕГО В JCRYPTOOL 11](#_Toc470171205)

[3.1. Настройка Eclipse IDE 11](#_Toc470171206)

[3.1.1. Настройка рабочего пространства (workspace) 11](#_Toc470171207)

[3.1.2. Клонирование Git-репозиториев 11](#_Toc470171208)

[3.1.3 Установка Eclipse Tycho 18](#_Toc470171209)

[3.1.4. Установка целевой платформы 19](#_Toc470171210)

[3.1.5. Запуск JCrypTool в Eclipse IDE 20](#_Toc470171211)

[3.2. Создание пользовательского плагина и внедрение его в JCrypTool 21](#_Toc470171212)

[3.2.1. Создание простого RCP-плагина 21](#_Toc470171213)

[3.2.2. Добавление нового плагина в JCrypTool 30](#_Toc470171214)

[3.2.3. Обновление конфигурации запуска (run configuration) 31](#_Toc470171215)

[3.2.4. Добавление расширений 33](#_Toc470171216)

[ЗАКЛЮЧЕНИЕ 35](#_Toc470171217)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc470171218)

# ВВЕДЕНИЕ

С 1 января 2016 года вступил в действие стандарт ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры», в котором описываются блочные шифры «Кузнечик» и «Магма».

Изучение стандарта может вызвать трудности у начинающего, поэтому целью данной работы является создание приложения, наглядно демонстрирующего работу криптографического алгоритма «Кузнечик».

Для достижения поставленной цели можно воспользоваться средствами JCrypTool – платформы для обучения криптографии, позволяющей студентам, преподавателям и разработчикам применять и изучать криптографические алгоритмы с помощью простого в использовании приложения. Платформа расширяется с помощью криптографических плагинов, которые могут быть добавлены любым пользователем.

Поскольку JCrypTool широко применяется во всем мире, использовать приложение, демонстрирующее работу блочного шифра «Кузнечик», смогут не только студенты нашего университета, но и любой человек, интересующийся криптографией.

# 1. ПОСТАНОВКА ЗАДАЧИ

Для достижения цели были поставлены следующие задачи:

* изучить работу алгоритма блочного шифрования «Кузнечик»;
* разработать и изобразить графически алгоритм работы демонстрационного приложения;
* создать плагин, демонстрирующий работу блочного шифра «Кузнечик» и внедрить его в JCrypTool.

На основании задач был составлен план работ:

# До 29.12.2016 г.:

* 1. Ознакомиться с алгоритмом блочного шифрования «Кузнечик» и разработать блок-схемы, отображающие работу демонстрационного приложения.
  2. Изучить возможности создания простого плагина и его внедрения в JCrypTool на локальном компьютере.

# До 25.05.2017 г.:

* 1. Более детально ознакомиться с работой алгоритма блочного шифрования «Кузнечик» и написать исходный код демонстрационного приложения.
  2. Интегрировать созданный криптографический плагин в общедоступный репозиторий JCrypTool.

# 2. ЛОГИКА ПРИЛОЖЕНИЯ



Схема 1 Приложение

На схеме 1 показан алгоритм приложения, демонстрирующего работу блочного шифра «Кузнечик».

Шаг 1: Ввод пользователем блока текста, который нужно зашифровать, и ключа

Шаг 2: Выбор между выводом полученного шифртекста без отображения промежуточных преобразований (перейти на шаг 3) и подробным отображением работы алгоритма (перейти на шаг 5)

Шаг 3: Развертывание ключа

Шаг 4: Зашифрование. Перейти на шаг 12

Шаг 5: Развертывание ключа

Шаг 6: Выбор между отображением всех этапов процедуры развертывания ключа (перейти на шаг 7) и простым выводом полученных итерационных ключей (перейти на шаг 8)

Шаг 7: Отображение каждого этапа развертывания ключа

Шаг 8: Отображение результата развертывания ключа

Шаг 9: Зашифрование

Шаг 10: Выбор между отображением всех этапов процедуры зашифрования (перейти на шаг 11) и простым выводом результата зашифрования (перейти на шаг 12)

Шаг 11: Отображение каждого этапа зашифрования

Шаг 12: Отображение результата зашифрования



Схема 2 Подпрограмма Развертывание ключа

На схеме 2 показана процедура развертывания ключа. Сначала вычисляют итерационные константы . На следующем шаге ключ шифрования разделяется на два итерационных ключа, после чего в цикле вычисляется еще четыре пары итерационных ключей.



Схема 3 Подпрограмма Зашифрование

На схеме 3 показана процедура зашифрования. Предполагается, что пользователю будут подробно показаны преобразования, которые выполняются в первом раунде зашифрования, далее со второго по девятый раунд будут отображены только конечные результаты, а десятый раунд, поскольку он отличается от остальных, также будет рассмотрен подробно.

# 3. СОЗДАНИЕ ПЛАГИНА И ВНЕДРЕНИЕ ЕГО В JCRYPTOOL

## 3.1. Настройка Eclipse IDE

Загрузите и установите:

1. Java SE Runtime Environment (JRE) или Java SE Development Kit (JDK) версии 8 или выше;
2. Eclipse Neon (пакет Eclipse for RCP and RAP Developers версии 4.6.x);
3. плагин EGit/JGit.

3.1.1. Настройка рабочего пространства (workspace)

Запустите Eclipse IDE с пустым workspace. Откройте Window → **Preferences** и выберите страницу **Workspace** в секции **General**. В группе Text file encoding выберите кодировку UTF-8.

Выберите страницу Compiler в секции Java. В выпадающем списке Java compliance level выберите 1.8.

3.1.2. Клонирование Git-репозиториев

Необходимо клонировать следующие репозитории:

* core: <https://github.com/jcryptool/core>
* crypto: <https://github.com/jcryptool/crypto>
* tests: <https://github.com/jcryptool/tests>
* doc: <https://github.com/jcryptool/doc>

На стартовой странице Eclipse IDE выберите Checkout Projects from Git (рис. 3.1).

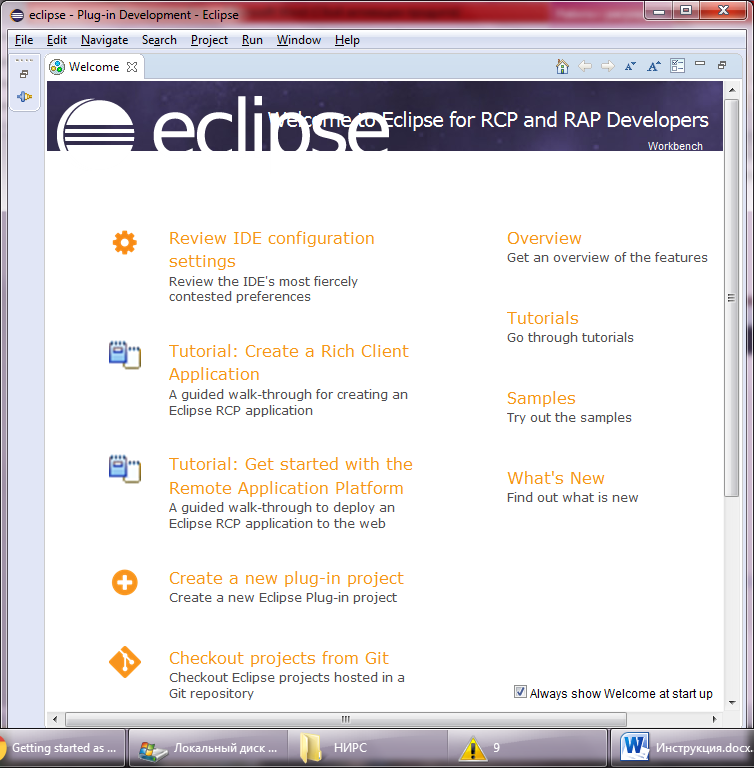


Рисунок 3.1 Стартовая страница Eclipse IDE

Откроется приложение Import Projects from Git (рис. 3.2). Выберите Clone URI и нажмите Next.

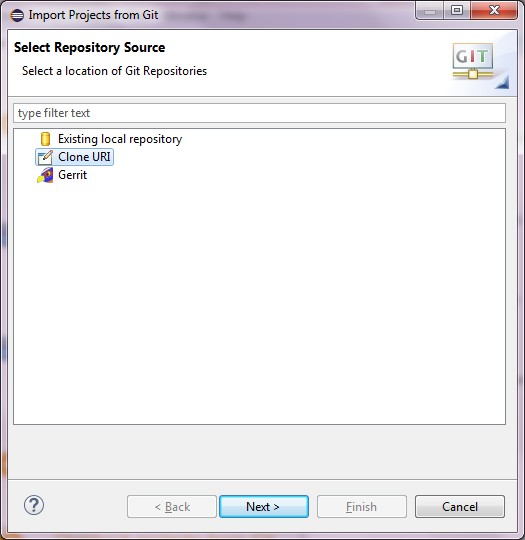


Рисунок 3.2 Import Projects from Git

В открывшемся окне (рис. 3.3) введите URI репозитория core, а также логин и пароль пользователя GitHub, остальные поля заполнятся автоматически. Нажмите Next.

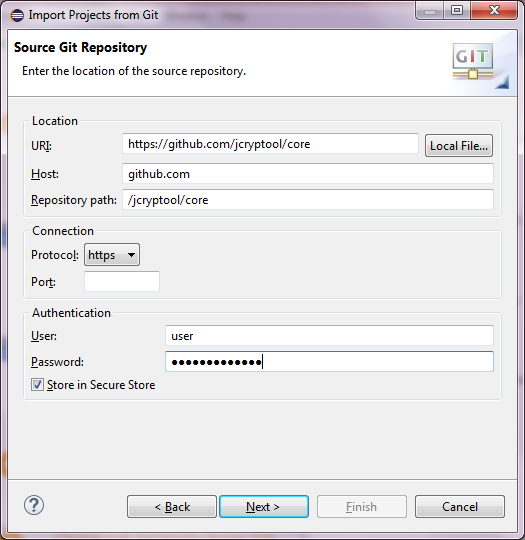


Рисунок 3.3

В открывшемся окне (рис. 3.4) выберите обе ветки и нажмите Next.

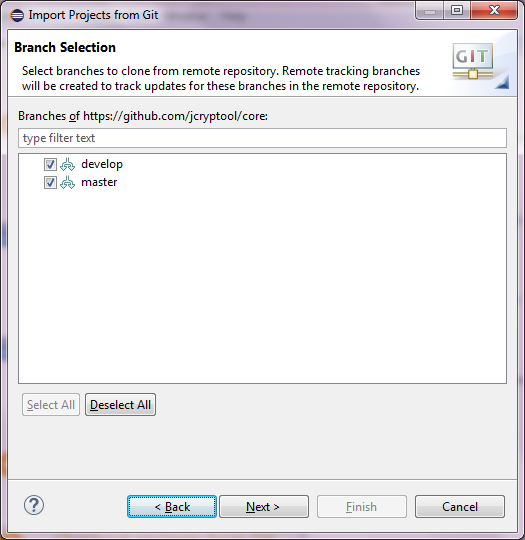


Рисунок 3.4

В открывшемся окне (рис. 3.5) введите директорию для хранения Git-репозитория на вашем компьютере. Не используйте текущее рабочее пространство Eclipse для хранения Git-репозитория. Нажмите Next.

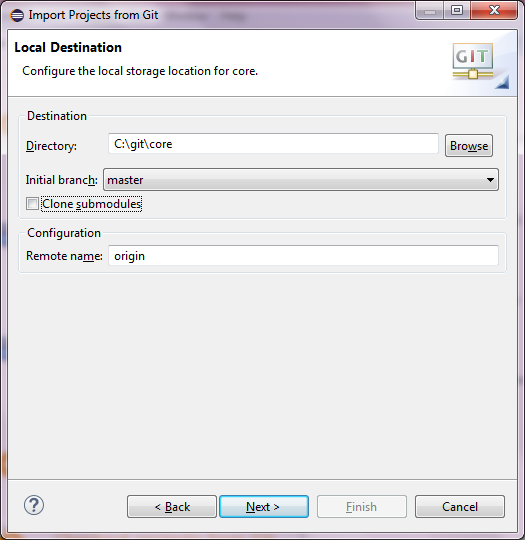


Рисунок 3.5

В открывшемся окне (рис. 3.6) оставьте отмеченным пункт **Import existing Eclipse projects** и нажмите **Next**.

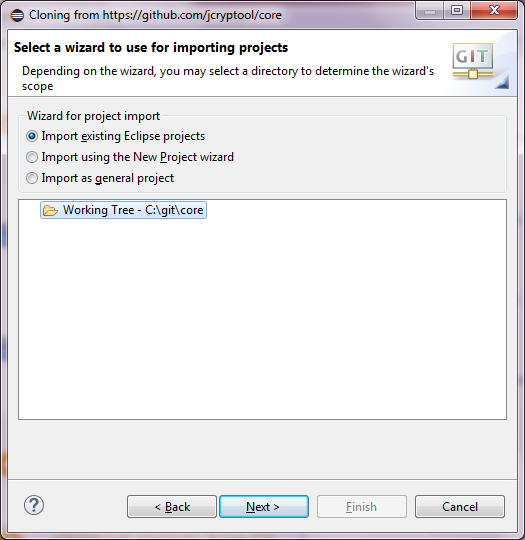


Рисунок 3.6

В открывшемся окне (рис. 3.7) выберите все проекты и нажмите **Finish**.

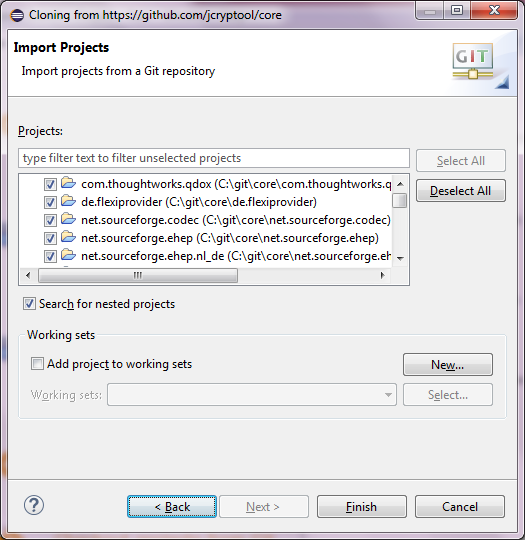


Рисунок 3.7

Итак, все проекты из core-репозитория добавлены и теперь доступны в вашем рабочем пространстве. Повторите те же шаги для остальных репозиториев.

3.1.3 Установка Eclipse Tycho

Сборка JCrypTool основана на Eclipse Tycho, для которого требуется Maven (M2E) и Eclipse Tycho Connector. Если Maven не установлен, можно загрузить его с помощью Eclipse Update Site. Установка Eclipse Tycho Connector требует нескольких дополнительных шагов.

Откройте файл org.jcryptool.product/pom.xml с помощью Maven POM Editor (рис. 3.8) и кликните на сообщение об ошибке, затем нажмите на ссылку Discover new m2e connectors. В результате этого запустится Eclipse MarketPlace и откроется Tycho Configurator. Установите его и перезапустите Eclipse, после этого ошибка в org.jcryptool.product/pom.xml исчезнет.

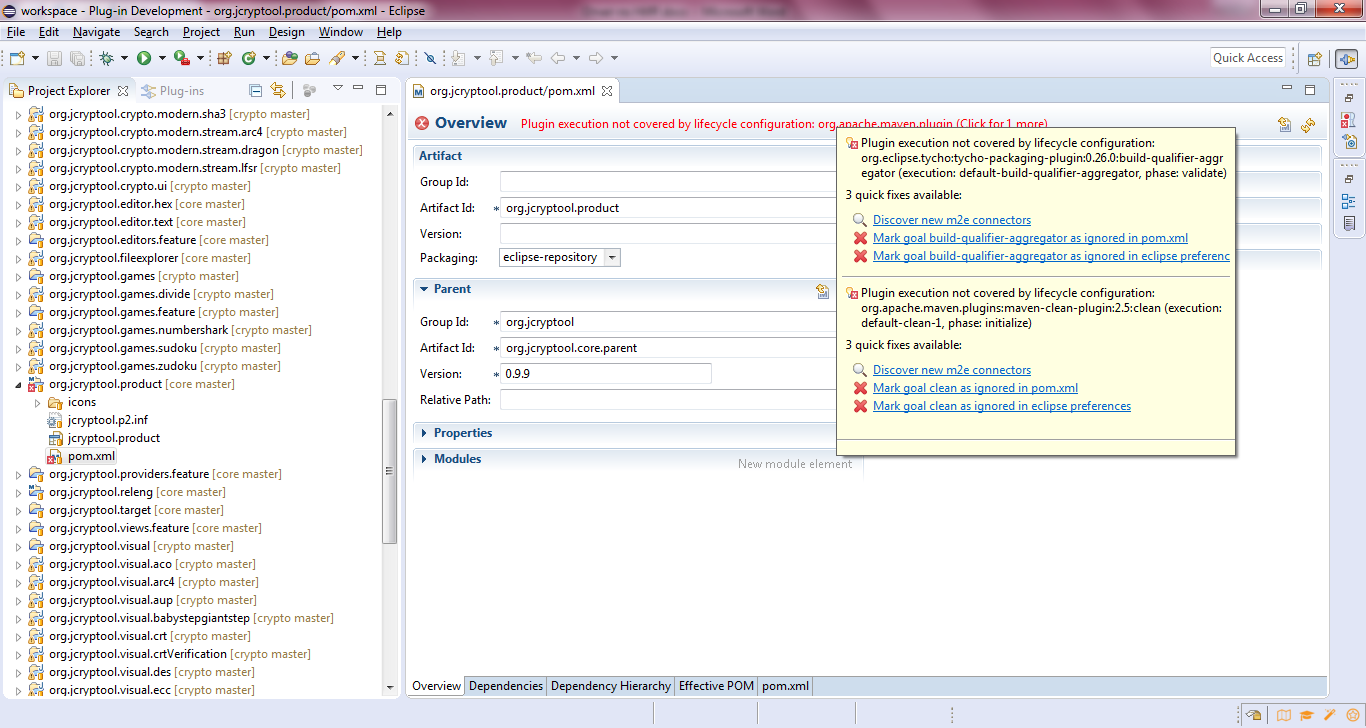


Рисунок 3.8

3.1.4. Установка целевой платформы

1) Откройте файл org.jcryptool.target.target (рис. 3.9) и в поле Locations нажмите клавишу Reload.

2) Кликните на ссылку Set as Target Platform, расположенную в правом верхнем углу.

После выполнения этих шагов не должно остаться ошибок сборки.

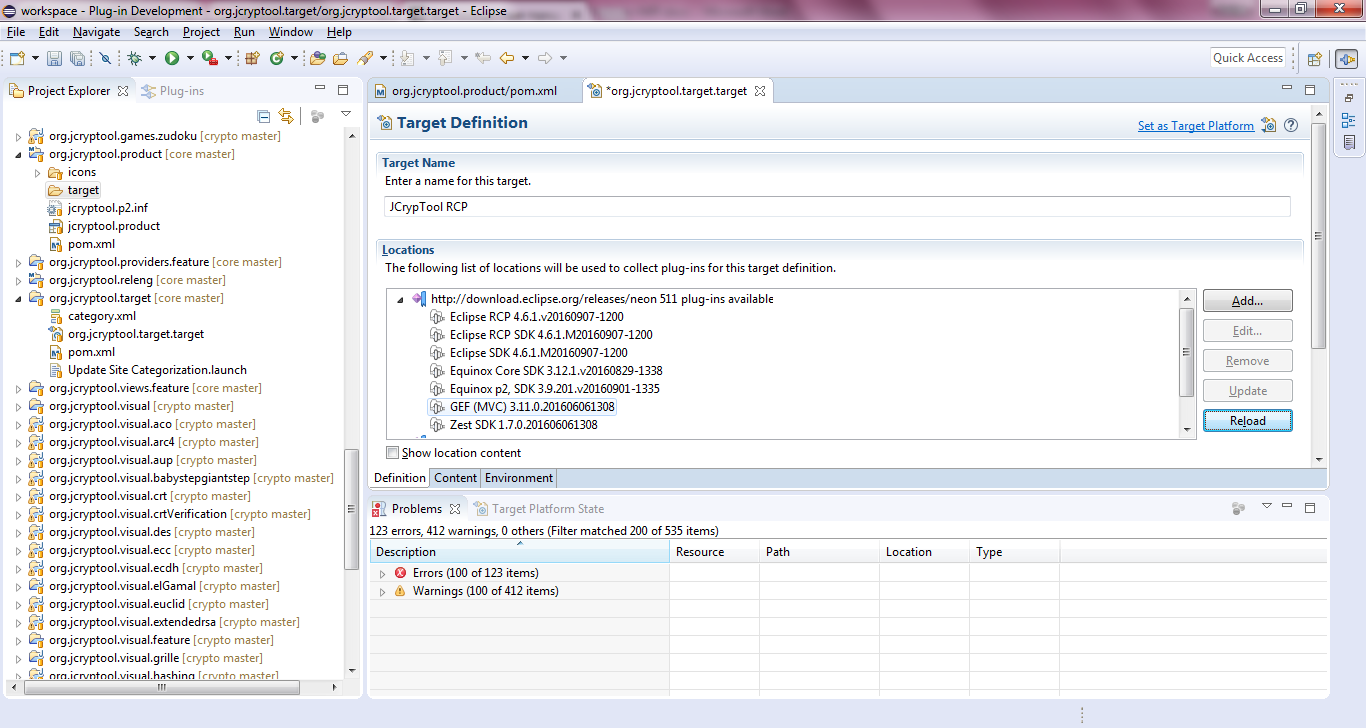


Рисунок 3.9

3.1.5. Запуск JCrypTool в Eclipse IDE

Откройте файл jcryptool.product, расположенный в проекте org.jcryptool.product, с помощью Product Configuration Editor (рис. 3.10). Кликните по ссылке Launch an Eclipse application в левом нижнем углу страницы. Это запустит JCrypTool.

Первый запуск может завершиться сообщением об ошибке. Чтобы это исправить, откройте Run → Run Configurations и выберите jcryptool.product в разделе Eclipse Application. Переключитесь на вкладку Plug-ins и нажмите на кнопку Add required plug-ins для добавления зависящих от платформы плагинов в вашу конфигурацию запуска. Этот шаг может потребоваться только при первом запуске.

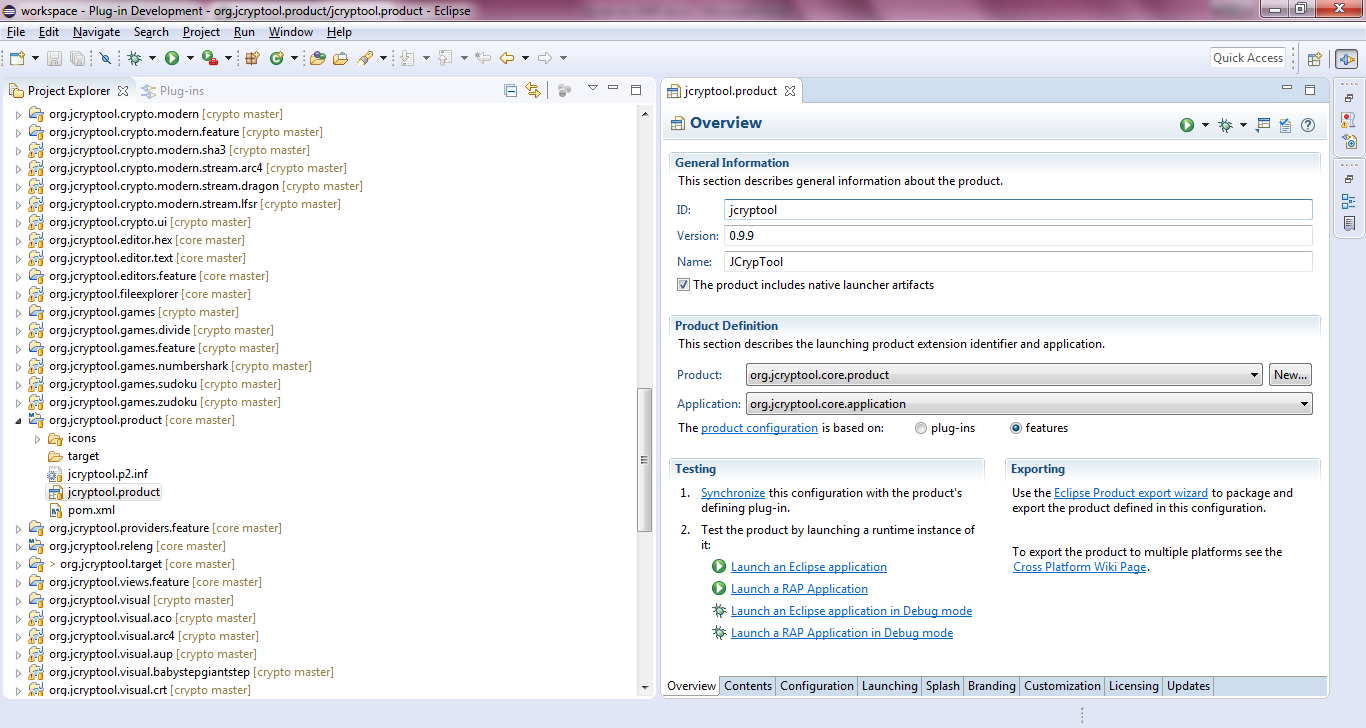


Рисунок 3.10

## 3.2. Создание пользовательского плагина и внедрение его в JCrypTool

3.2.1. Создание простого RCP-плагина

Запустите File → New → Plug-in Project. В открывшемся окне введите имя плагина (рис 3.11). Нажмите Next.

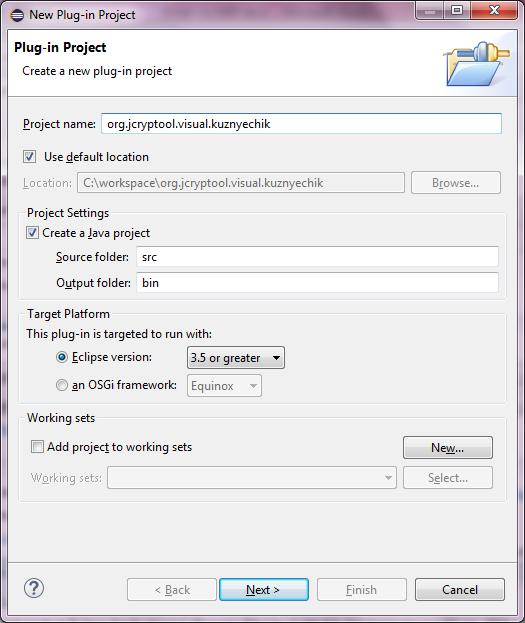


Рисунок 3.11

В открывшемся окне (рис. 3.12) выберите Yes в качестве ответа на вопрос «Would you like to client a rich client application?» и нажмите Next.

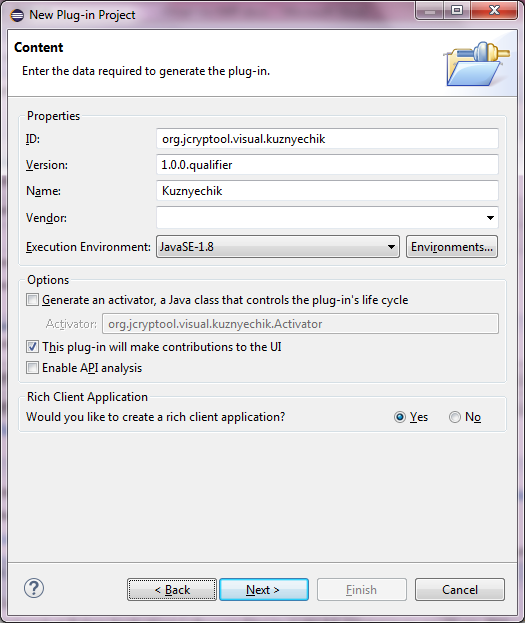


Рисунок 3.12

В открывшемся окне (рис. 3.13) выберите шаблон RCP 3.x application (minimal) и нажмите Next.

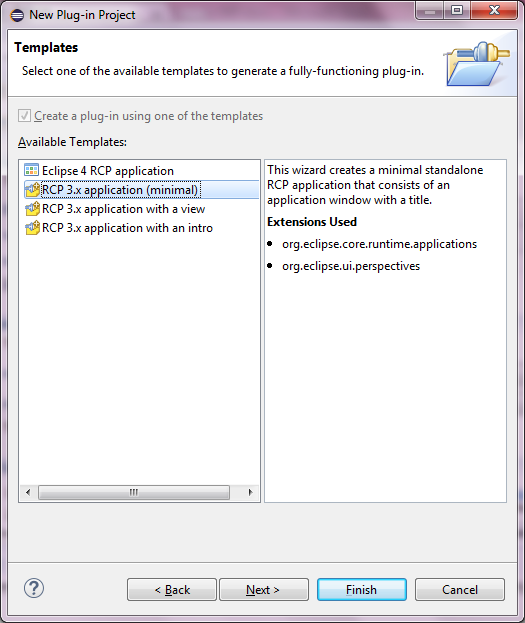


Рисунок 3.13

В открывшемся окне введите заголовок для окна приложения и введите имя класса приложения, затем нажмите Finish.

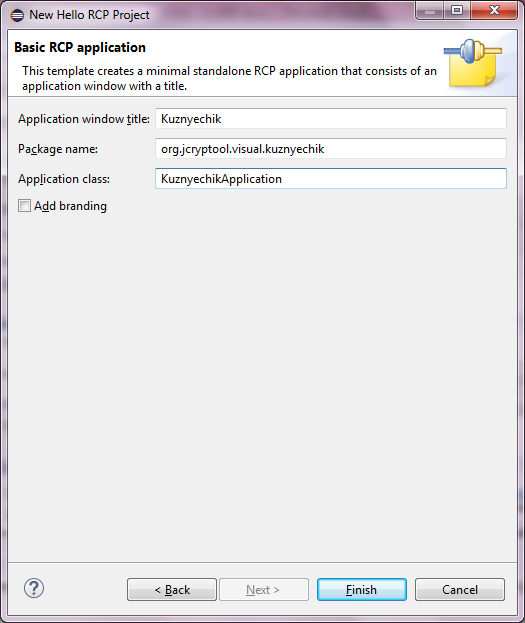


Рисунок 3.14

Откроем MANIFEST.MF и запустим приложение. В результате отображается пустое окно (рис. 3.15).

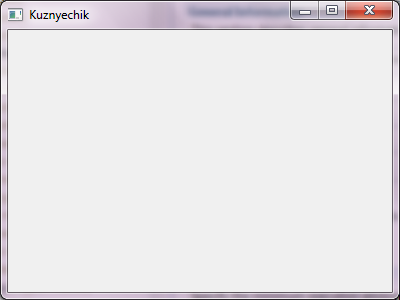


Рисунок 3.15

Теперь нужно добавить вид (view). Для этого снова перейдем к файлу MANIFEST.MF и откроем вкладку Extensions (рис. 3.16).

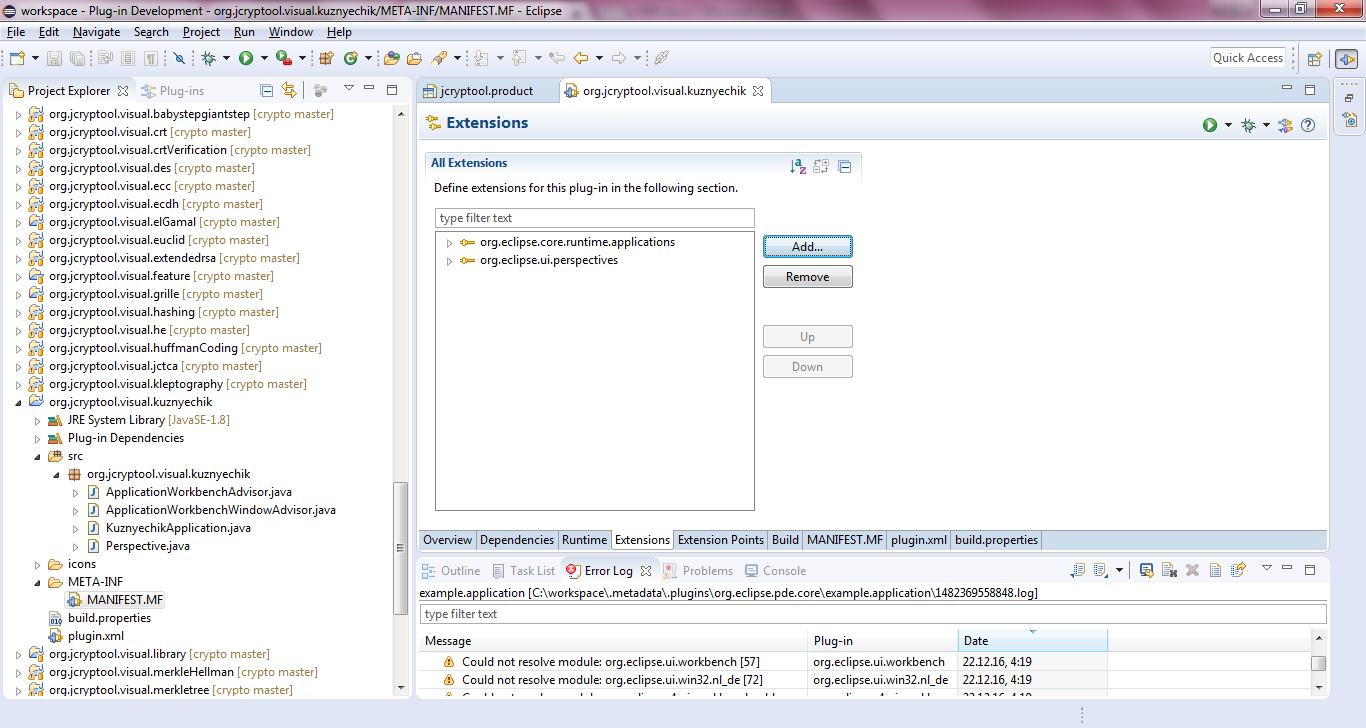


Рисунок 3.16

Нажмем клавишу Add и выберем точку расширения org.eclipse.ui.views (рис. 3.17). В списке расширений должно появиться еще одно расширение. Щелкнем правой кнопкой мыши по добавленному расширению и выберем в контекстном меню New → view. Настройки для добавленного элемента отображаются в правой части окна (рис. 3.18).

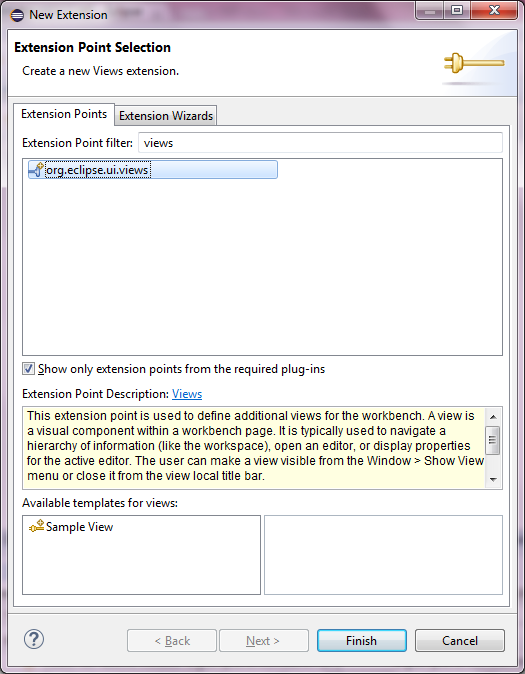


Рисунок 3.17

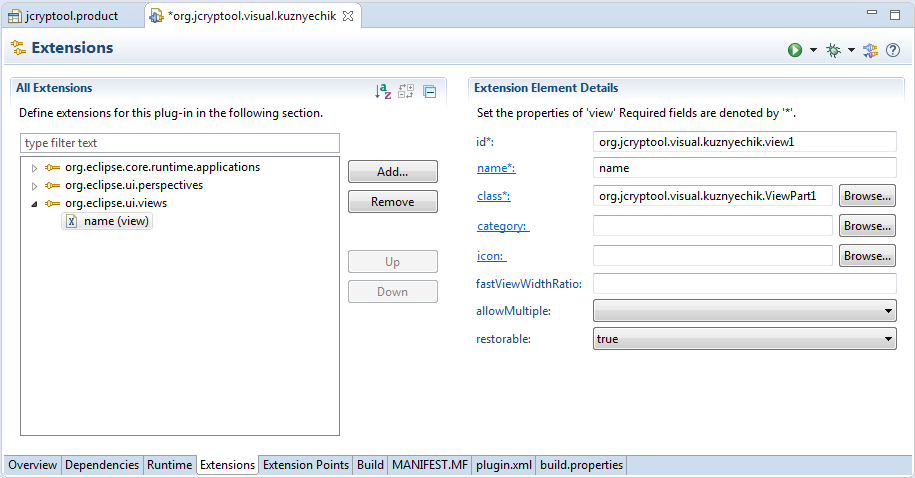


Рисунок 3.18

Создадим класс для добавленного вида, для этого кликнем по ссылке class\*. Дадим классу название MainView (рис. 3.19).

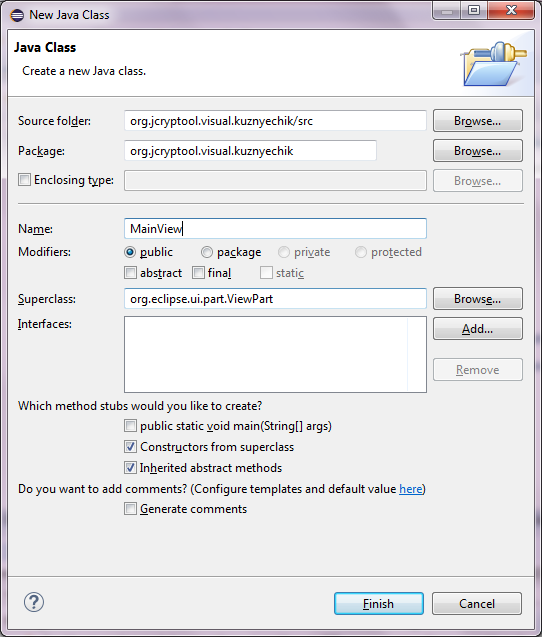


Рисунок 3.19

После создания класса должен открыться файл MainView.java. Для наглядности можно, к примеру, задать изображение в качестве фона. Ниже приведен листинг.

Листинг 3.1 MainView.java

**package** org.jcryptool.visual.kuznyechik;

**import** org.eclipse.swt.widgets.Composite;

**import** org.eclipse.ui.part.ViewPart;

**import** org.eclipse.swt.graphics.Image;

**public** **class** MainView **extends** ViewPart {

**public** MainView() {

// **TODO** Auto-generated constructor stub

}

@Override

**public** **void** createPartControl(Composite parent) {

// **TODO** Auto-generated method stub

Image image = **new** Image (**null**, "C://Kuznyechik.jpg");

parent.setBackgroundImage(image);

}

@Override

**public** **void** setFocus() {

// **TODO** Auto-generated method stub

}

}

3.2.2. Добавление нового плагина в JCrypTool

Необходимо добавить созданный плагин в соответствующий feature. Для этого нужно открыть feature.xml в org.jcryptool.visual.feature, переключиться на вкладку Including Plug-ins, нажать клавишу Add, в открывшемся окне выбрать необходимый плагин и сохранить изменения (рис. 3.20).

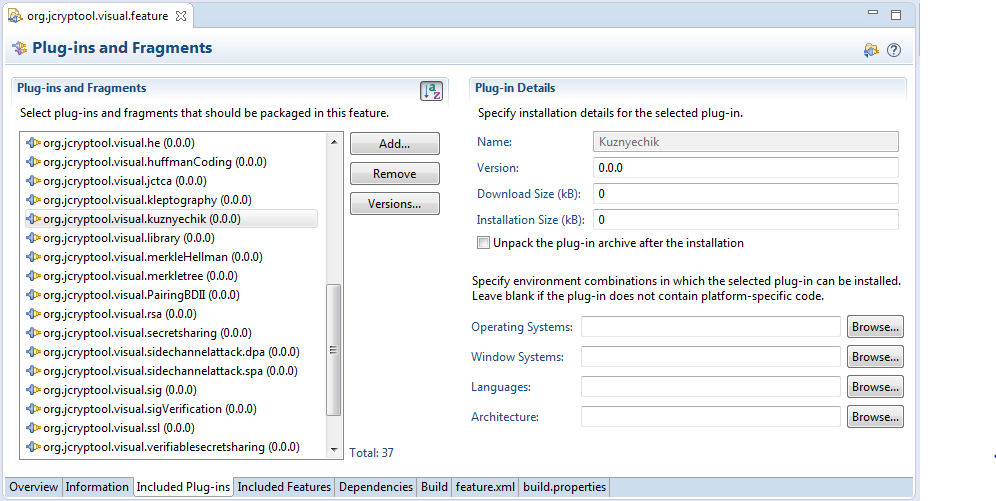


Рисунок 3.20

3.2.3. Обновление конфигурации запуска (run configuration)

Для запуска созданного плагина в JCrypTool необходимо изменить конфигурацию запуска. Нужно запустить Run Configurations и выбрать JCrypTool, переключить на вкладку Plug-ins, в секции Workspace найти необходимый плагин и добавить его (рис. 3.21). Затем нужно нажать кнопку Apply и запустить JCrypTool.

Теперь плагин доступен в JCryptool. Чтобы убедиться в этом, в JCrypTool нужно открыть Help → About, в открывшемся окне About JCrypTool нажать на кнопку Installation Details и выбрать вкладку Plug-ins. В открывшейся таблице (рис. 3.22) можно найти добавленный плагин.

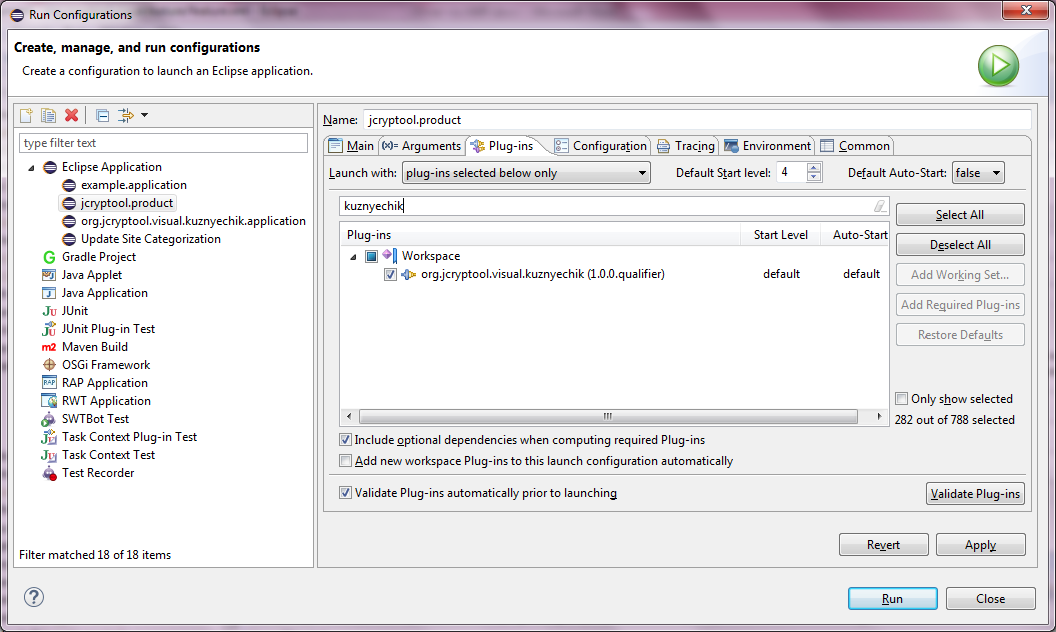


Рисунок 3.21

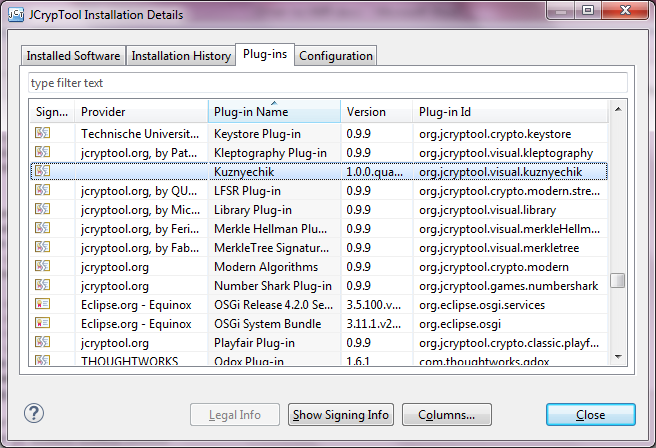


Рисунок 3.22

3.2.4. Добавление расширений

Чтобы плагин был доступен через меню JCrypTool, необходимо добавить новое расширение. Нужно открыть plugin.xml, который находится в org.jcryptool.visual.kuznyechik, и добавить в нем следующий код:

<extension point="org.jcryptool.core.operations.visuals">

<Visual

name="Kuznyechik"

viewId="org.jcryptool.visual.kuznyechik.MainView"

contextHelpId="org.jcryptool.visual.kuznyechik.MainView">

</Visual>

</extension>

Теперь нужно запустить JCrypTool и убедиться, что плагин доступен в меню Visuals (рис. 3.23).

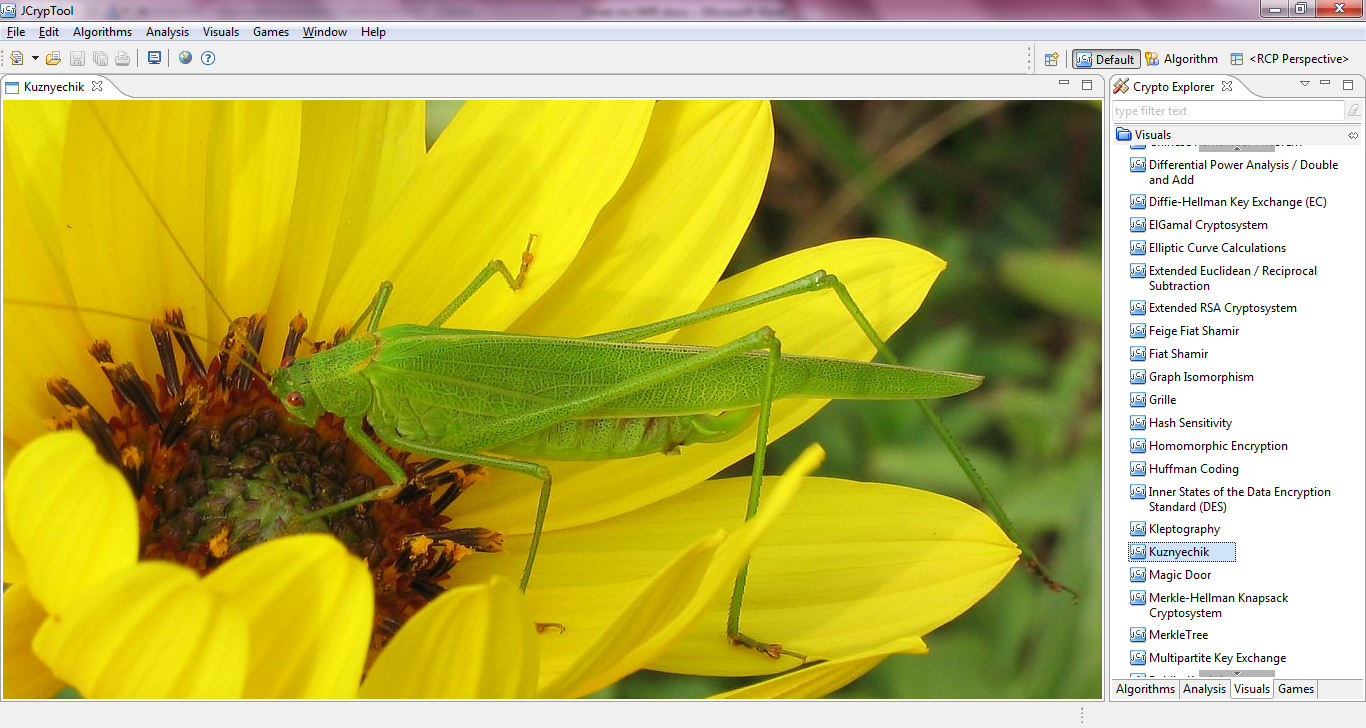


Рисунок 3.23 Созданный плагин внедрен в JCrypTool

# ЗАКЛЮЧЕНИЕ

В процессе выполнения научно-исследовательской работы на данном этапе получены следующие результаты:

1. На основе ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры» разработан алгоритм работы демонстрационного приложения для блочного шифра «Кузнечик», построены блок-схемы, отображающие все этапы алгоритма.
2. Изучены способы создания плагинов. Созданный плагин внедрен в JCrypTool на локальном компьютере.

Нужно отметить, что задачи, выделенные для реализации на первом этапе, выполнены в срок.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры». М: Стандартинформ, 2015.
2. Getting started as a JCrypTool Developer // GitHub. URL: <https://github.com/jcryptool/core/wiki/Getting-started-as-a-JCrypTool-Developer> (дата обращения: 01.10.2016).
3. Creating a new Plug in // GitHub. URL: <https://github.com/jcryptool/core/wiki/Creating-a-new-Plug-in> (дата обращения: 01.10.2016).
4. Extension Points // GitHub. URL: <https://github.com/jcryptool/core/wiki/Extension-Points> (дата обращения: 01.10.2016).
5. Разработка простого Eclipse RCP приложения // Хабрахабр. URL: <https://habrahabr.ru/post/139340/> (дата обращения: 20.11.2016)