National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Dmitry Cherniy
Student ID: 19230265

School of Computing
National College of Ireland

Supervisor: Ross Spelman

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Dmitry Cherniy……

Student ID: 19230265……

Programme: Cybersecurity…… **Year:** 2021……

Module: MSc Internship……

Lecturer: Ross Spelman ……

Submission Due Date: 16/08/2021……

Project Title: Securing Embedded Metadata with Symmetric and Asymmetric Encryption ……

Word Count: 1498…… **Page Count:** 9……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: 16/08/2021……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Dmitry Cherniy
Student ID: 19230265

1 Introduction

The configuration manual describes the research project, the essential methods, practical implementations, and evaluations that have taken place. It proposes a new design to secure embedded metadata using the combination of symmetric and asymmetric encryption algorithms. The project consists of six different methods to encrypt and decrypt the metadata. It is practically implemented in Java programming language using the Netbeans-8.2 IDE, the Java security APIs and Bouncy Castle Crypto APIs. The project can be configured on a stand-alone computer or in a virtual environment. The use of a virtual machine is not compulsory, but it helps create a better-isolated environment for testing and evaluation purposes. System configuration begins with installing Ubuntu Linux along with the required programming environment and additional software tools. The following chapters will describe the configuration process in more detail.

2 System Configuration

Host machine:

Base machine: Mac OS 11.4
System type: 64-bit operating system
Processor: 4.2 GHz Quad-Core Intel Core i7
Memory: 40 GB 2400 MHz DDR4
GPU: Radeon Pro 580, 8 GB
Storage: SSD: 100 GB of free space

Guest machine:

Hypervisor: VMware Fusion Player, Version 12.1.0
Virtual Machine: Linux Ubuntu 20.04.1 LTS
System type: 64-bit operating system
Processor: Two processor cores
Memory: 4 GB System
Type: 64 bit
Virtual Disk: 7.2 GB of free space

2.1 Download and Instal Virtual Machine Software

Firstly, the hypervisor software must be downloaded and installed. We used VMware Fusion Player; the latest version can be found here:
https://my.vmware.com/web/vmware/evalcenter?p=fusion-player-personal.
Next, we need to download the Virtual Machine image file. Ubuntu 20.04.1 LTS can be downloaded from the following link:
http://old-releases.ubuntu.com/releases/20.04.1/ubuntu-20.04-beta-desktop-amd64.iso

2.2 Install Ubuntu Linux

First, we need to create and configure a new virtual machine from the downloaded image in VMware Fusion. Alternatively, the required software can be installed on a stand-alone computer.
We set up Ubuntu Linux with the "ubuntu" username. If a different username is chosen, the path to the configuration files and working directory must be adjusted accordingly.

3 Setting up the Programming Environment

In this section, we discuss how to set up and configure the programming environment for our project.

3.1 Setting up Working directory

The working directory for our project is /home/ubuntu/Research Project/TEST. All the input files for metadata encryption must be placed in that folder (Figure 1).

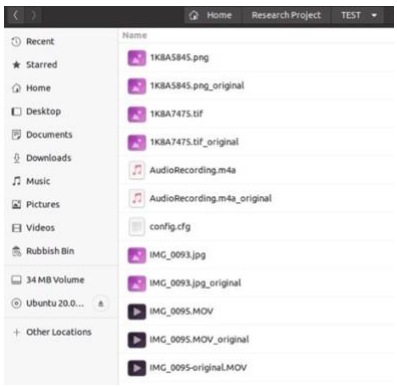


Figure 1: Project Working Directory

3.2 Download and install Java JDK and NetBeans

JDK 8 is bundled with NetBeans 8.2 and distributed by Oracle.
The bundle can be downloaded and installed in the terminal using wget command-line tool:
$ wget -c http://download.oracle.com/otn-pub/java/jdk-nb/8u111-8.2/jdk-8u111-nb-8_2-linux-x64.sh
Make downloaded file executable, run the installer script and follow the onscreen instructions:
$ chmod +x jdk-8u111-nb-8_2-linux-x64.sh
$ sh jdk-8u111-nb-8_2-Linux-x64.sh

Figure 2: JDK 8 and NetBeans 8.2 Bundle Installation

3.3 Download, install and configure ExifTool

Exiftool can be installed in the terminal using the command line:
$ sudo apt-get install exiftool
Exiftool is used with SystemCommandExecutor Java class which executes system commands from a Java application. We also need to create an exiftool configuration file *config.cfg* and place it in the working directory: */home/ubuntu/Research Project/TEST* directory:

```
# NOTE: All tag names used in the following tables are case sensitive.

# The %Image::ExifTool::UserDefined hash defines new tags to be added
# to existing tables.
%Image::ExifTool::UserDefined = (

    # XMP tags may be added to existing namespaces:
    'Image::ExifTool::XMP::xmp' => {
        # Example 5.  XMP-xmp:Senders_public_key
        Senders_public_key => { Groups => { 2 => 'Author' } },
        # add more user-defined XMP-xmp tags here...
        Enc_metadata => { Groups => { 2 => 'Author' } },
        Sym_pass_with_send_pub => { Groups => { 2 => 'Author' } },
        Sym_pass_with_rec_pub => { Groups => { 2 => 'Author' } },
        Hash => { Groups => { 2 => 'Author' } },
        Signature => { Groups => { 2 => 'Author' } },
    },
config.cfg
```

Figure 3: ExifTool Configuration File

3.4 Download and Add Java Libraries

Apache common codec is required for base64 operations; it can be downloaded from: https://repository.apache.org/content/repositories/releases/commons-codec/commons-codec/1.9/
Bouncy Castle Crypto APIs is a software library that introduces a wide range of cryptographic functionality necessary for our project. It can be downloaded from: https://www.bouncycastle.org/fr/download/jce-jdk13-168.jar

4 Configuration and Execution

In this section, we discuss how to import Java code into NetBeans and run the project.

4.1 Import Research Project

Import researchProject.zip into NetBeans: File-> Import Project -> From zip.
NetBeans will report “broken reference” error because we need to add jar files downloaded earlier.
Right click on an imported project -> Properties -> Libraries:

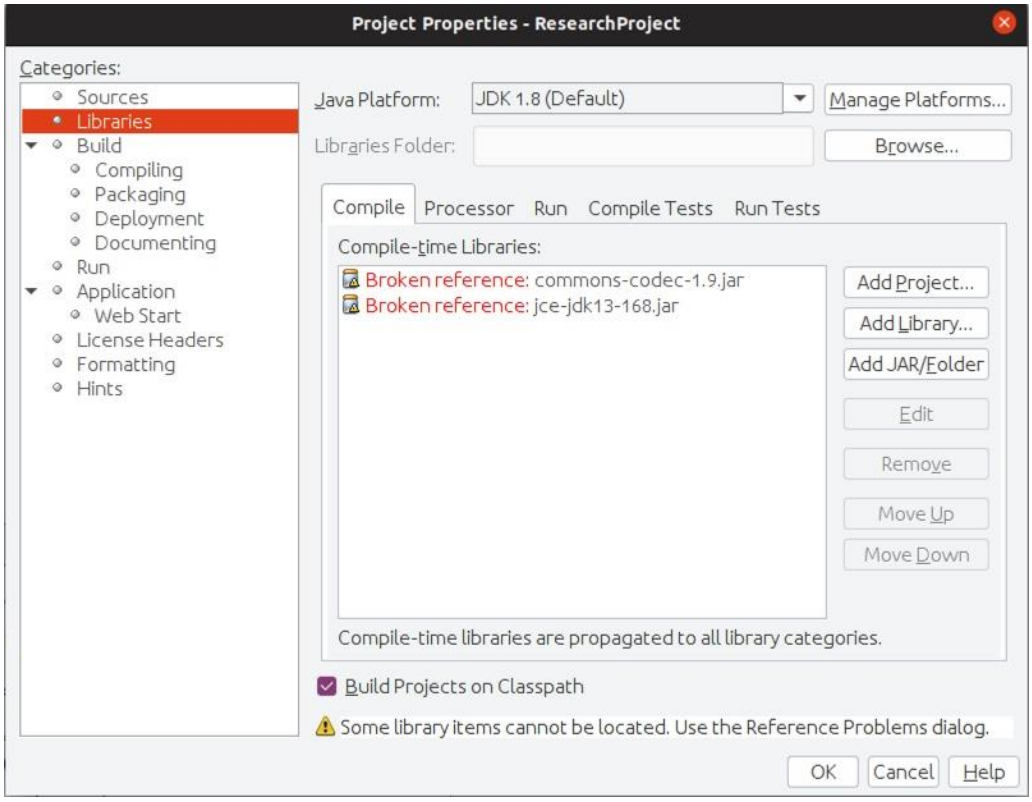


Figure 4: Project Import

Click Add JAR/Folder and navigate to downloaded commons-codec-1.9.jar and jce-jdk-168.jar libraries and click OK button to save configuration:

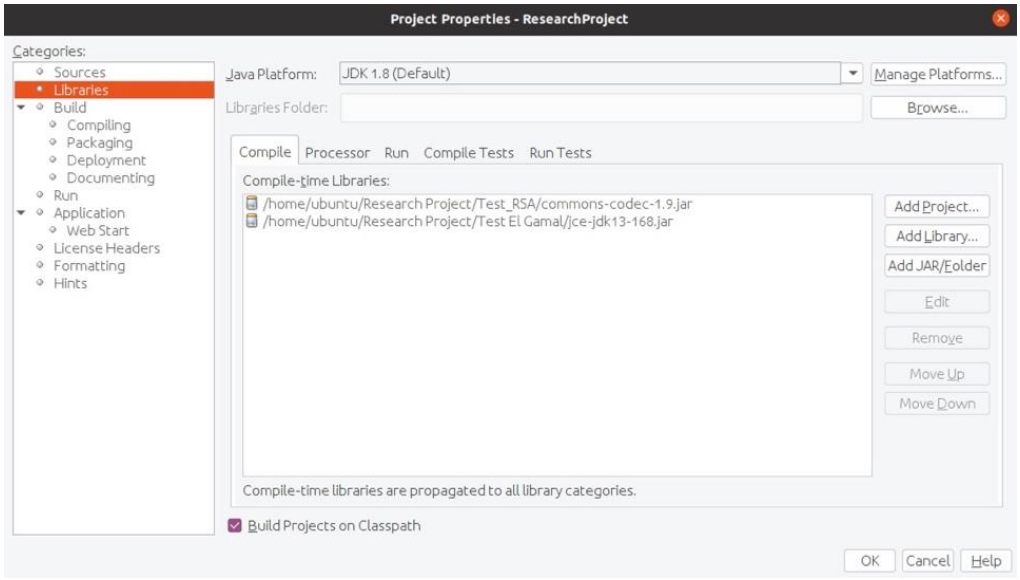


Figure 5: Adding Java Libraries

4.2 Create public/private key pairs

Before we can run the project, we need to generate public/private key pairs. To create DSA keys, right-click on GenerateDsaKeys.java -> Run file. NetBeans will show "BUILD SUCCESSFUL" output, and generated keys will be placed in ../ResearchProject/DSA folder:

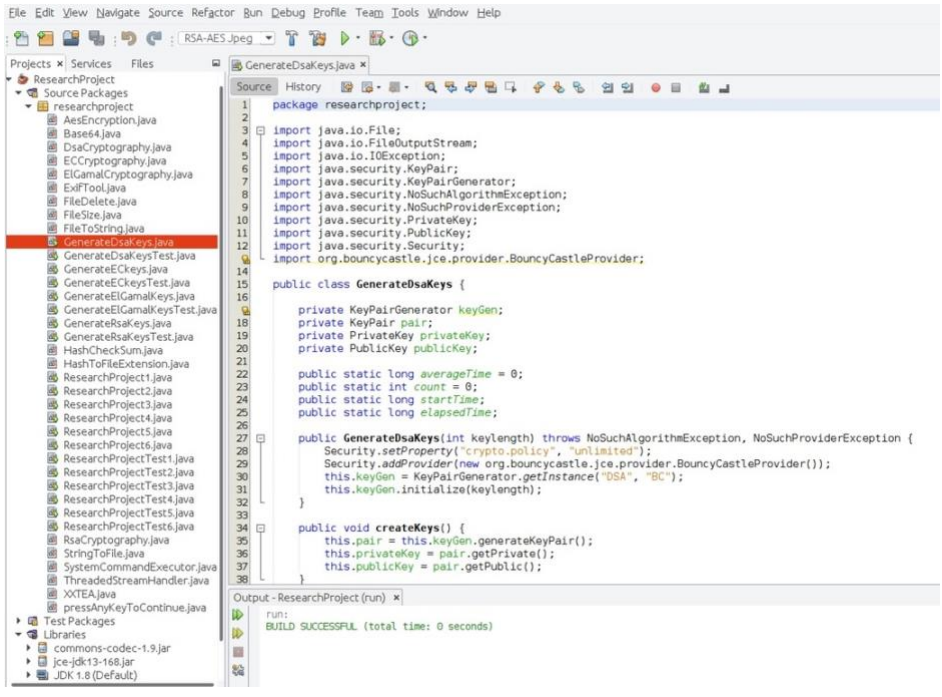


Figure 6: Generating Key Pairs

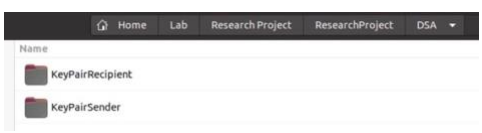


Figure 7: Created Key Pairs

ECDSA keys are created by running GenerateECkeys.java class.

To generate ElGamal keys, run GenerateElGamalKeys.java class, and for RSA keys, run GenerateRsaKeys.java.

4.3 Verifying the Path to ExifTool Configuration File

ExifTool.java class contains variable *config*, which specifies the location of the *config.cfg* file. Make sure that the file *config.cfg* exists, and the path to it is correct in line 81:

```
79
80    public static void addTags (String path, String tag, String value)throws IOException, InterruptedException {
81      String config="/home/ubuntu/Research\\ Project/TEST/config.cfg";
        List<String> commands = new ArrayList<String>();
83      commands.add("/bin/sh");
84      commands.add("-c");
```

Figure 8: Path to the *config*.cfg file

4.4 Configure Arguments for Netbeans

To set arguments (the locations of input files) we need select Run -> Set Project Configuration -> Customize. Select New and provide the Configuration's name, the Main Class's name (ResearchProject1.java – ResearchProject6.java) in the dialogue box, and an absolute path to the input file. Hit OK to save the properties:

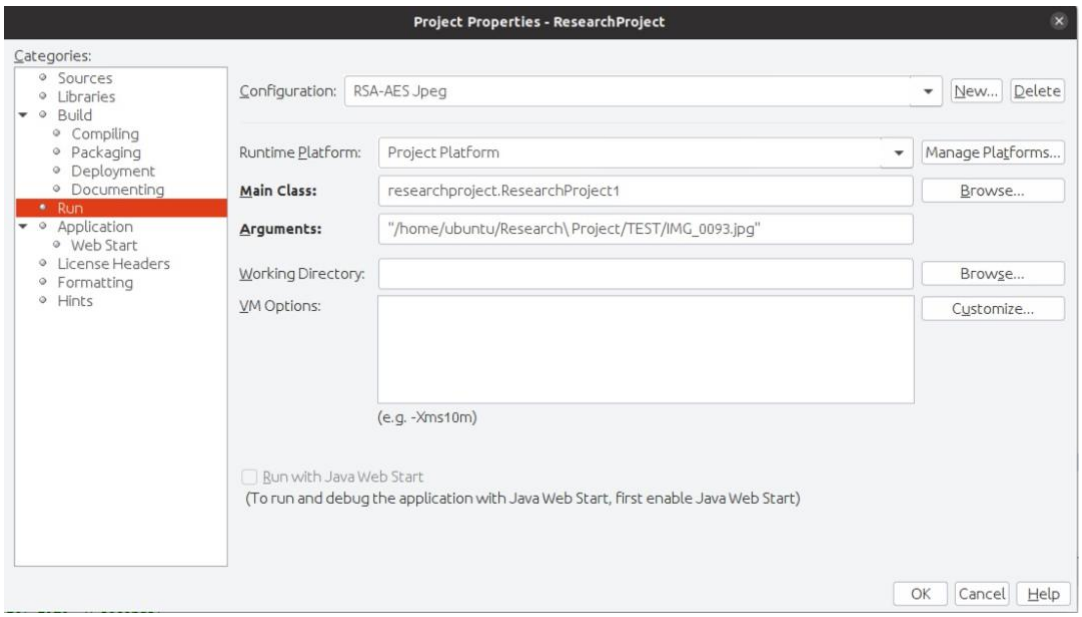


Figure 9: NetBeans Arguments Configuration

This procedure must be repeated for the remaining methods (ResearchProject1.java – ResearchProject6.java) and arguments (input files). Specifications of ResearchProject1.java – ResearchProject6.java methods are shown in Table 1.

Table 1. Specifications of the Proposed Methods

Method #	Symmetric Key Algorithm	Symmetric Key Size	Public Key Algorithm	Public Key Type / Size	Digital Signature	Digital Signature Key Type / Size
ResearchProject1.java	AES	128bits	RSA	RSA / 2048 bits	RSA	RSA / 2048 bits
ResearchProject2.java	XXTEA	128bits	RSA	RSA / 2048 bits	RSA	RSA 2048/ bits
ResearchProject3.java	AES	128bits	ElGamal	ElGamal / 2048 bits	DSA	DSA / 2048 bits
ResearchProject4.java	XXTEA	128bits	ElGamal	ElGamal / 2048 bits	DSA	DSA / 2048 bits

ResearchProject5.java	AES	128bits	ECDH	ECDSA Curve P-256 / 256 bits	ECDSA	ECDSA P-256 / 256 bits
ResearchProject6.java	XXTEA	128bits	ECDH	ECDSA Curve P-256 / 256 bits	ECDSA	ECDSA P-256 / 256 bits

4.5 Running the Project

The project methods run by selecting configuration name from the drop-down menu created earlier and pressing the Start button:

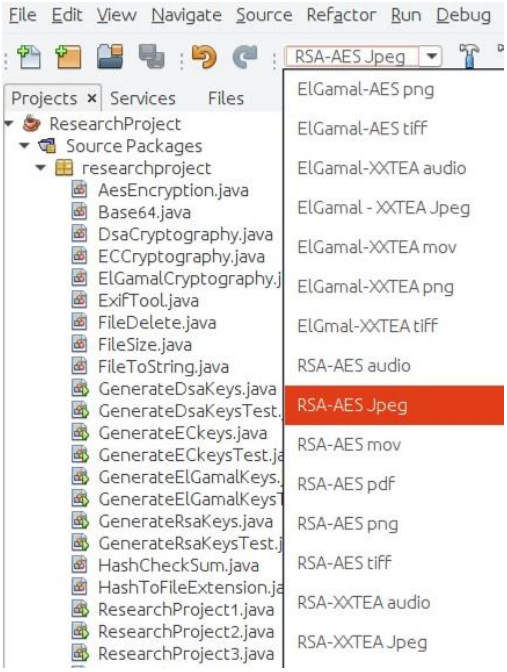


Figure 10: Running the Project

The selected method will encrypt the embedded metadata in the given file, than program pauses, waiting for user input:

Figure 11: Embedded Metadata Encryption Process

After the user hits any key, the program will continue and decrypts the metadata:

Figure 12: Embedded Metadata Decryption Process

To view the embedded metadata, we can use ExifTool:

$ exiftool filename

Figure 13: Encrypted Metadata

4.6 Testing the Project

We have created additional Java classes for testing and evaluation purposes: GenerateRsaKeysTest.java, GenerateElGamalKeysTest.java, GenerateECKeysTest.java, GenerateDsaKeysTest.java. These methods are used to test the average key pairs generation times. Methods run the number of times specified by the final value of variable *i* in the *for* loop (Figure 14, line 14), which can be adjusted to the required value.

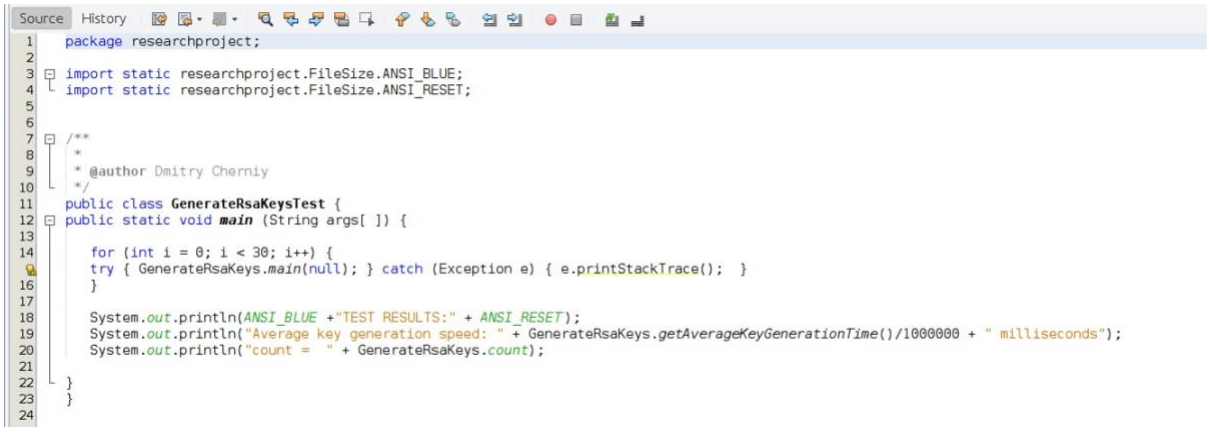
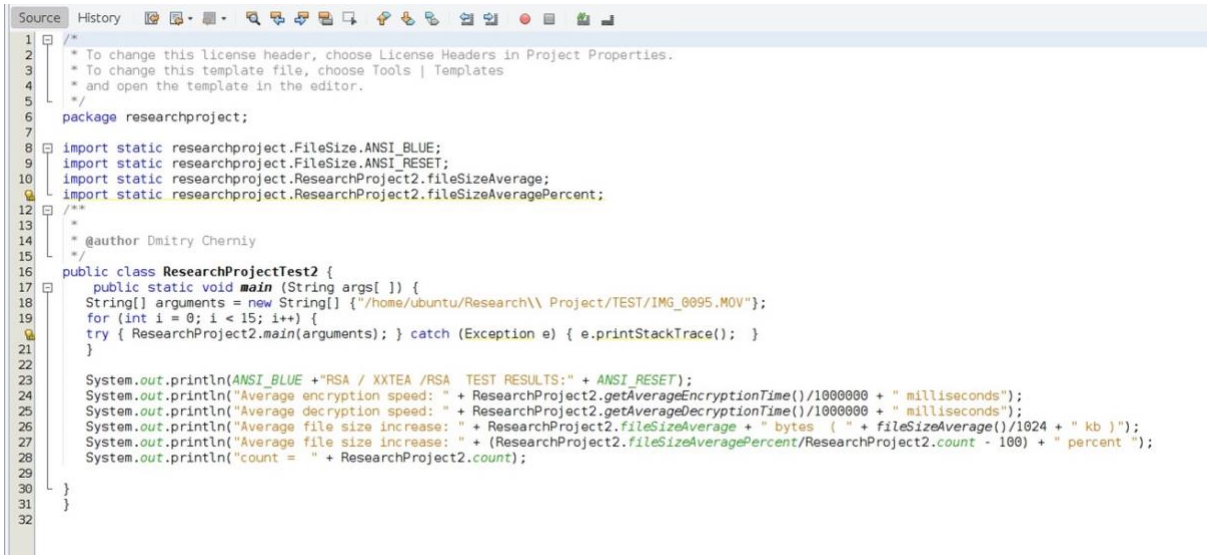


Figure 14: GenerateRsaKeysTest.java

To test and evaluate the primary metrics of the metadata encryption/decryption methods, we created ResearchProjectTest1.java, ResearchProjectTest2.java, ResearchProjectTest3.java, ResearchProjectTest4.java, ResearchProjectTest5.java, ResearchProjectTest6.java classes. The number of iterations (test runs) can be set by changing variable *i* in the *for*-loop line 19, Figure 15. For the detailed specification of the above methods, refer to Table 1. For testing purposes, we need to comment line 165 for methods ResearchProjectTest1.java – ResearchProjectTest4.java and line 156 for methods ResearchProjectTest5.java and ResearchProjectTest6.java (Figure 16). Otherwise, the program will stop and wait for user input. The program computes average encryption/decryption times and files size differences before/after encryption. Memory utilisation is calculated separately inside each method.



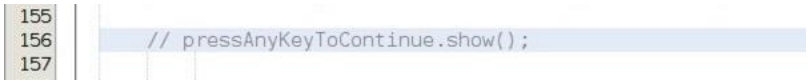
```
Source  History
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package researchproject;

import static researchproject.FileSize.ANSI_BLUE;
import static researchproject.FileSize.ANSI_RESET;
import static researchproject.ResearchProject2.fileSizeAverage;
import static researchproject.ResearchProject2.fileSizeAveragePercent;
/**
 *
 * @author Dmitry Cherniy
 */
public class ResearchProjectTest2 {
    public static void main (String args[ ]) {
   String[] arguments = new String[] {"/home/ubuntu/Research\\ Project/TEST/IMG_0095.MOV"};
   for (int i = 0; i < 15; i++) {
   try { ResearchProject2.main(arguments); } catch (Exception e) { e.printStackTrace();  }
   }

   System.out.println(ANSI_BLUE +"RSA / XXTEA /RSA  TEST RESULTS:" + ANSI_RESET);
   System.out.println("Average encryption speed: " + ResearchProject2.getAverageEncryptionTime()/1000000 + " milliseconds");
   System.out.println("Average decryption speed: " + ResearchProject2.getAverageDecryptionTime()/1000000 + " milliseconds");
   System.out.println("Average file size increase: " + ResearchProject2.fileSizeAverage + " bytes  ( " + fileSizeAverage()/1024 + " kb )");
   System.out.println("Average file size increase: " + (ResearchProject2.fileSizeAveragePercent/ResearchProject2.count - 100) + " percent ");
   System.out.println("count =  " + ResearchProject2.count);

}
}
```

Figure 15: Testing the Project



```
155
156     // pressAnyKeyToContinue.show();
157
```

Figure 16: ResearchProjectTest1.java