**Министерство образования Республики Беларусь**

**Учреждение образования**

**БелорусскиЙ государственный университет**

**информатики и радиоэлектроники**

**Факультет компьютерных систем и сетей**

**Кафедра программного обеспечения информационных технологий**

**Отчет по лабораторной работе № 5**

**по предмету**

**Надежность программного обеспечения на** **тему:**

**Работа в среде автоматизированного тестирования TestComplete**

|  |  |  |
| --- | --- | --- |
| **Выполнил**  **Студент гр. 051001** |  | Кравченко А. А. |
| **Проверил** |  | Деменковец Д.В. |

**Минск, 2022**

В ходе данной лабораторной работы было протестировано приложение для постановки ЭЦП и ее проверки по алгоритму DSA. В качестве параметров для алгоритма были взяты значения, сохраненные в виде таблицы Excel (для каждого теста собственный лист).

**Тесты**

1. **input\_check** (тестирование введения некорректных параметров)
   1. **TestNoParameters —** тестирование постановки и проверки подписи при отсутствии параметров
   2. **TestCharParameters —** тестирование постановки и проверки подписи при заполнении полей не числовыми значениями
   3. **TestPNotPrime** **—** тестирование постановки и проверки подписи при не удовлетворении условия простоты параметра p
   4. **TestQNotPrime** **—** тестирование постановки и проверки подписи при не удовлетворении условия простоты параметра q
   5. **TestHInterval** **—** тестирование постановки и проверки подписи при не удовлетворении условия принадлежности h интервалу (1, p - 1)
   6. **TestTestQNotP\_1** **—** тестирование постановки и проверки подписи при не удовлетворении условия делимости (p - 1) на q
   7. **TestXInterval** **—** тестирование постановки подписи при не удовлетворении условия принадлежности x интервалу (0, q)
   8. **TestKInterval** — тестирование постановки подписи при не удовлетворении условия принадлежности k интервалу (0, q)
2. **signature\_check** (тестирование постановки и проверки подписи)
   1. **TestEmptyFile —** тестирование постановки и проверки подписи при удовлетворении параметрами всех условий в случае работы с пустым файлом
   2. **TestBSUIR —** тестирование постановки и проверки подписи при удовлетворении параметрами всех условий в случае работы с файлом, содержащем текст «BSUIR»
   3. **TestWrongFormat —** тестирование проверки подписи при передаче файла без подписи
   4. **TestWrongSignature —** тестирование проверки подписи при некорректной подписи

**Скрипты модулей**

**1. DDTWork** (модуль для работы с DDT)

def get\_values(sheet\_name, values):

DDT.ExcelDriver("..\Resources\TestSequence.xlsx", sheet\_name)

result = {}

for value in values:

result[value] = DDT.CurrentDriver.Value[value]

DDT.CloseDriver(DDT.CurrentDriver.Name)

return result

**2. TestSamples** (модуль с шаблонами для проведении тестирования)

import DDTWork

def startDSA():

TestedApps.Dsa.Run(1)

def stopDSA():

TestedApps.Dsa.Close()

def test\_parameters\_sign(excel\_sheet, alies, file=False):

if file:

res = DDTWork.get\_values(excel\_sheet, ["p", "q", "h", "x", "k", "file"])

else:

res = DDTWork.get\_values(excel\_sheet, ["p", "q", "h", "x", "k"])

DSA = alies

frmDSA = DSA.frmDsa

frmDSA.rbSign.Select()

tb = frmDSA.tbP

tb.Text = res["p"]

tb = frmDSA.tbQ

tb.Text = res["q"]

tb = frmDSA.tbH

tb.Text = res["h"]

tb = frmDSA.tbX

tb.Text = res["x"]

tb = frmDSA.tbK

tb.Text = res["k"]

frmDSA.gbRadiobtnSign.btnChoice.Click()

if file:

filepath = aqFileSystem.ExpandFileName("..\Resources\TestFiles\\"+res["file"])

test\_choose\_file(filepath, alies)

def test\_parameters\_check\_sign(excel\_sheet, alies, file\_s=False):

if file\_s:

res = DDTWork.get\_values(excel\_sheet, ["p", "q", "h", "y", "file\_s"])

else:

res = DDTWork.get\_values(excel\_sheet, ["p", "q", "h", "y"])

DSA = alies

frmDSA = DSA.frmDsa

frmDSA.rbCheckSign.Select()

tb = frmDSA.tbP

tb.Text = res["p"]

tb = frmDSA.tbQ

tb.Text = res["q"]

tb = frmDSA.tbH

tb.Text = res["h"]

tb = frmDSA.tbY

tb.Text = res["y"]

frmDSA.gbRadiobtnSign.btnChoice.Click()

if file\_s:

filepath = aqFileSystem.ExpandFileName("..\Resources\TestFiles\\"+res["file\_s"])

test\_choose\_file(filepath, alies)

def test\_dlg\_warning(static, static\_wnd\_caption, alies):

def test\_choose\_file(filepath, alies):

dlg = alies.dlg\_2

dlg.cbx\_2.ComboBox.Edit.SetText(filepath)

dlg.btn\_.ClickButton()

def test\_sign\_values(sheet\_excel, alies):

DSA = alies

values\_check = DDTWork.get\_values(sheet\_excel, ["y", "g", "hash", "r", "s"])

res = aqString.Compare(DSA.frmDsa.tbY.Text, values\_check["y"], True)

if res != 0:

Log.Error("y is incorrect")

res = aqString.Compare(DSA.frmDsa.tbG.Text, values\_check["g"], True)

if res != 0:

Log.Error("g is incorrect")

res = aqString.Compare(DSA.frmDsa.tbR.Text, values\_check["r"], True)

if res != 0:

Log.Error("r is incorrect")

res = aqString.Compare(DSA.frmDsa.tbS.Text, values\_check["s"], True)

if res != 0:

Log.Error("s is incorrect")

res = aqString.Compare(DSA.wndDSA.gbSignResult.tbHM.Text, values\_check["hash"], True)

if res != 0:

Log.Error("H(M) is incorrect")

def test\_check\_sign\_values(sheet\_excel, alies):

DSA = alies

values\_check = DDTWork.get\_values(sheet\_excel, ["g", "hash", "r", "s", "w", "v"])

res = aqString.Compare(DSA.frmDsa.tbG.Text, values\_check["g"], True)

if res != 0:

Log.Error("g is incorrect")

res = aqString.Compare(DSA.frmDsa.tbHMC.Text, values\_check["hash"], True)

if res != 0:

Log.Error("H(M') is incorrect")

res = aqString.Compare(DSA.frmDsa.tbRC.Text, values\_check["r"], True)

if res != 0:

Log.Error("r is incorrect")

res = aqString.Compare(DSA.frmDsa.tbSC.Text, values\_check["s"], True)

if res != 0:

Log.Error("s is incorrect")

res = aqString.Compare(DSA.frmDsa.tbW.Text, values\_check["w"], True)

if res != 0:

Log.Error("w is incorrect")

res = aqString.Compare(DSA.frmDsa.tbV.Text, values\_check["v"], True)

if res != 0:

Log.Error("v is incorrect")

**3. Tests** (тестирование некорректно введенных параметров)

import TestSamples

def test\_no\_sequence\_sign():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("KeyNone", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static, "Введите", DSA)

def test\_no\_parameters\_check\_sign():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("KeyNone", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static, "Введите", DSA)

def test\_chars\_sign():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("Chars", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static2, "не числовое значение", DSA)

def test\_chars\_check\_sign():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("Chars", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static2, "не числовое значение", DSA)

def test\_sign\_p\_not\_prime():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("p\_not\_prime", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static3, "p не простое число", DSA)

def test\_check\_sign\_p\_not\_prime():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("p\_not\_prime", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static3, "p не простое число", DSA)

def test\_sign\_q\_not\_prime():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("q\_not\_prime", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static4, "q не простое число", DSA)

def test\_check\_sign\_q\_not\_prime():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("q\_not\_prime", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static4, "q не простое число", DSA)

def test\_sign\_q\_not\_p\_1():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("q\_p-1", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static5, "q не является делителем (p - 1)", DSA)

def test\_check\_sign\_q\_not\_p\_1():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("q\_p-1", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static5, "q не является делителем (p - 1)", DSA)

def test\_sign\_h\_not\_interval():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("h\_interval", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static6, "h не в интервале (1, p - 1)", DSA)

def test\_check\_sign\_h\_not\_interval():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("h\_interval", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static6, "h не в интервале (1, p - 1)", DSA)

def test\_sign\_x\_not\_interval():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("x\_interval", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static7, "x не в интервале (0, q)", DSA)

def test\_sign\_k\_not\_interval():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("k\_interval", DSA)

TestSamples.test\_dlg\_warning(DSA.dlg\_.Static8, "k не в интервале (0, q)", DSA)

**4. TestSignature** (тестирование постановки и проверки подписи)

import TestSamples

def test\_sign\_empty\_file():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("emptyfile", DSA, True)

TestSamples.test\_sign\_values("emptyfile", DSA)

DSA.frmDsa.gbRadiobtnSign.btnSave.Click()

filepath = aqFileSystem.ExpandFileName("..\Resources\TestFiles\empty\_s.txt")

TestSamples.test\_choose\_file(filepath, DSA)

def test\_check\_sign\_empty\_file():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("emptyfile", DSA, True)

TestSamples.test\_check\_sign\_values("emptyfile", DSA)

aqObject.CheckProperty(DSA.frmDsa.lblResult, "Text", cmpContains, "Подпись верна")

def test\_sign\_BSUIR():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_sign("BSUIR", DSA, True)

TestSamples.test\_sign\_values("BSUIR", DSA)

DSA.frmDsa.gbRadiobtnSign.btnSave.Click()

filepath = aqFileSystem.ExpandFileName("..\Resources\TestFiles\BSUIR\_s.txt")

TestSamples.test\_choose\_file(filepath, DSA)

def test\_check\_sign\_BSUIR():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("BSUIR", DSA, True)

TestSamples.test\_check\_sign\_values("BSUIR", DSA)

aqObject.CheckProperty(DSA.frmDsa.lblResult, "Text", cmpContains, "Подпись верна")

def test\_wrong\_format\_check\_sign():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("wrong\_format", DSA, True)

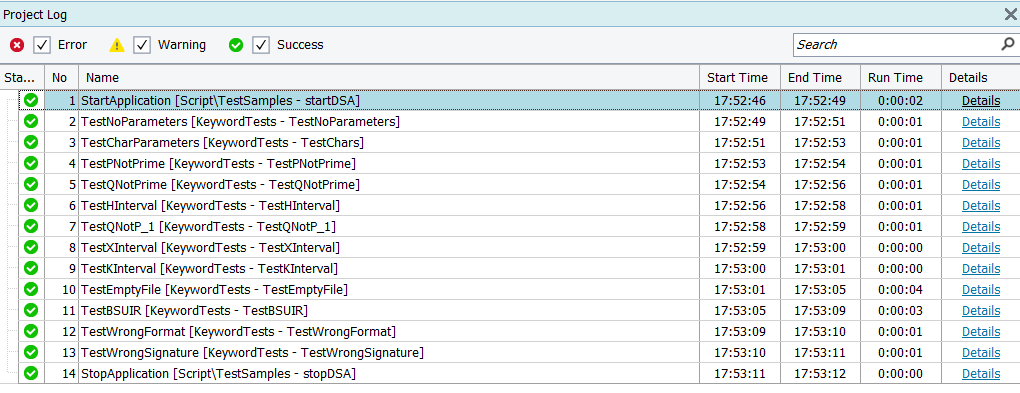
TestSamples.test\_dlg\_warning(DSA.dlg\_.Static9, "Подпись не может быть проверена", DSA)

def test\_wrong\_signature():

DSA = Aliases.Dsa

TestSamples.test\_parameters\_check\_sign("wrong\_signature", DSA, True)

aqObject.CheckProperty(DSA.frmDsa.lblResult, "Text", cmpContains, "Подпись не верна")

**Результаты тестирования**