﻿using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using SecurityLibrary;

namespace SecurityPackageTest

{

[TestClass]

public class RailFenceTest

{

string mainPlain1 = "meetmeaftertheparty";

string mainPlain2 = "meetmeafterthepartyxx";

string mainCipher = "mematrhpryetefeteat".ToUpper();

string mainCipher2 = "mtaehayemfrereettpt".ToUpper();

string mainCipher3 = "mtaehayemfrerxeettptx".ToUpper();

int mainKey = 2;

int mainKey2 = 3;

string newPlain = "nothingisasitseems";

string newCipher = "NTIGSSTEMOHNIAISES";

int newkey = 2;

[TestMethod]

public void RailFenceTestEnc1()

{

RailFence algorithm = new RailFence();

string cipher = algorithm.Encrypt(mainPlain1, mainKey);

Assert.IsTrue(cipher.Equals(mainCipher, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestDec1()

{

RailFence algorithm = new RailFence();

string plain = algorithm.Decrypt(mainCipher, mainKey);

Assert.IsTrue(plain.Equals(mainPlain1, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestAnalysis1()

{

RailFence algorithm = new RailFence();

int key = algorithm.Analyse(mainPlain1, mainCipher);

Assert.AreEqual(mainKey, key);

}

[TestMethod]

public void RailFenceTestEnc2()

{

RailFence algorithm = new RailFence();

string cipher = algorithm.Encrypt(mainPlain1, mainKey2);

// Add x's or not

Assert.IsTrue(cipher.Equals(mainCipher2, StringComparison.InvariantCultureIgnoreCase)

|| cipher.Equals(mainCipher3, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestDec2()

{

RailFence algorithm = new RailFence();

string plain1 = algorithm.Decrypt(mainCipher2, mainKey2);

string plain2 = algorithm.Decrypt(mainCipher3, mainKey2);

Assert.IsTrue(plain1.Equals(mainPlain1, StringComparison.InvariantCultureIgnoreCase)

|| plain2.Equals(mainPlain2, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestAnalysis2()

{

RailFence algorithm = new RailFence();

int key = algorithm.Analyse(mainPlain1, mainCipher2);

int key2 = algorithm.Analyse(mainPlain1, mainCipher3);

Assert.IsTrue(mainKey2 == key || mainKey2 == key2);

}

[TestMethod]

public void RailFenceTestNewEnc()

{

RailFence algorithm = new RailFence();

string cipher = algorithm.Encrypt(newPlain, newkey);

Assert.IsTrue(cipher.Equals(newCipher, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestNewDec()

{

RailFence algorithm = new RailFence();

string plain = algorithm.Decrypt(newCipher, newkey);

Assert.IsTrue(plain.Equals(newPlain, StringComparison.InvariantCultureIgnoreCase));

}

[TestMethod]

public void RailFenceTestNewAnalysis()

{

RailFence algorithm = new RailFence();

int key = algorithm.Analyse(newPlain, newCipher);

Assert.AreEqual(newkey, key);

}

}

}