Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**ЗВІТ**

**Про виконання лабораторної роботи № 4**

**з дисципліни «Безпека програм та даних»**

**«Створення програмної реалізації алгоритму шифрування з відкритим ключем RSA з використанням Microsoft CryptoAPI»**

**Лектор:**

доц. кафедри ПЗ

Сенів М. М.

**Виконав:**

студ. групи ПІ-41

Павленчик М. М.

**Прийняв:**

доц. кафедри ПЗ

Сенів М. М.

«\_\_\_» \_\_\_\_\_\_ 2018 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**ТЕМА РОБОТИ**: Створення програмної реалізації алгоритму шифрування з відкритим ключем RSA з використанням Microsoft CryptoAPI.

**МЕТА РОБОТИ**: Ознайомитись з методами і засобами криптографії з відкритим ключем, навчитись створювати програмні засоби з використанням криптографічних інтерфейсів.

### Теоретичні відомості

Концепція криптографії з відкритим ключем була запропонована Уітфілдом Діффі (Whitfield Diffie) та Мартіном Хеллманом (Martin Hellman), і, незалежно, Ральфом Мерклом (Ralph Merkle). Основна ідея – використовувати ключі парами, що складаються з ключа шифрування та ключа дешифрування, які неможливо обчислити один з одного.

Загальна схема виглядає наступним чином:

1. Кожен користувач генерує пару ключів: один для шифрування і один для дешифрування.
2. Кожен користувач публікує свій ключ шифрування, розміщує його у відкритому для всіх доступі. Другий ключ, парний до відкритого, зберігається в таємниці.
3. Якщо користувач А збирається надіслати повідомлення користувачеві В, він шифрує повідомлення відкритим ключем користувача В.
4. Коли користувач В отримує повідомлення, він дешифрує його за допомогою свого приватного ключа. Інший користувач не зможе дешифрувати повідомлення, оскільки приватний ключ В відомий тільки користувачеві В.

Алгоритми шифрування з відкритим ключем розроблялися для того, щоб вирішити дві найбільш важкі задачі, що виникли при використанні симетричного шифрування.

Першою задачею є розподіл ключа. При симетричному шифруванні потрібно, щоб обидві сторони вже мали спільний ключ, що якимось чином повинний бути їм заздалегідь переданий. Ця вимога заперечує всю суть криптографії, а саме можливість підтримувати загальну таємність при комунікаціях.

Другою задачею є необхідність створення таких механізмів, при використанні яких неможливо було б підмінити кого-небудь з учасників, тобто потрібен цифровий підпис.

Криптографія з відкритим ключем дозволяє вирішити набагато ширше коло задач, ніж криптографія класична. Однак існує ряд причин, з яких асиметричні алгоритми шифрування не можуть повноцінно замінити симетричні алгоритми.

**Завдання для лабораторної роботи**

З використання функцій CryptoAPI створити програмну реалізацію алгоритму шифрування RSA. Оцінити швидкість шифрування алгоритму RSA та порівняти її зі швидкістю шифрування алгоритму RC5, реалізованого в роботі № 3, зробити відповідні висновки та відобразити їх у звіті до лабораторної роботи.

### Результати

Протокол роботи програми

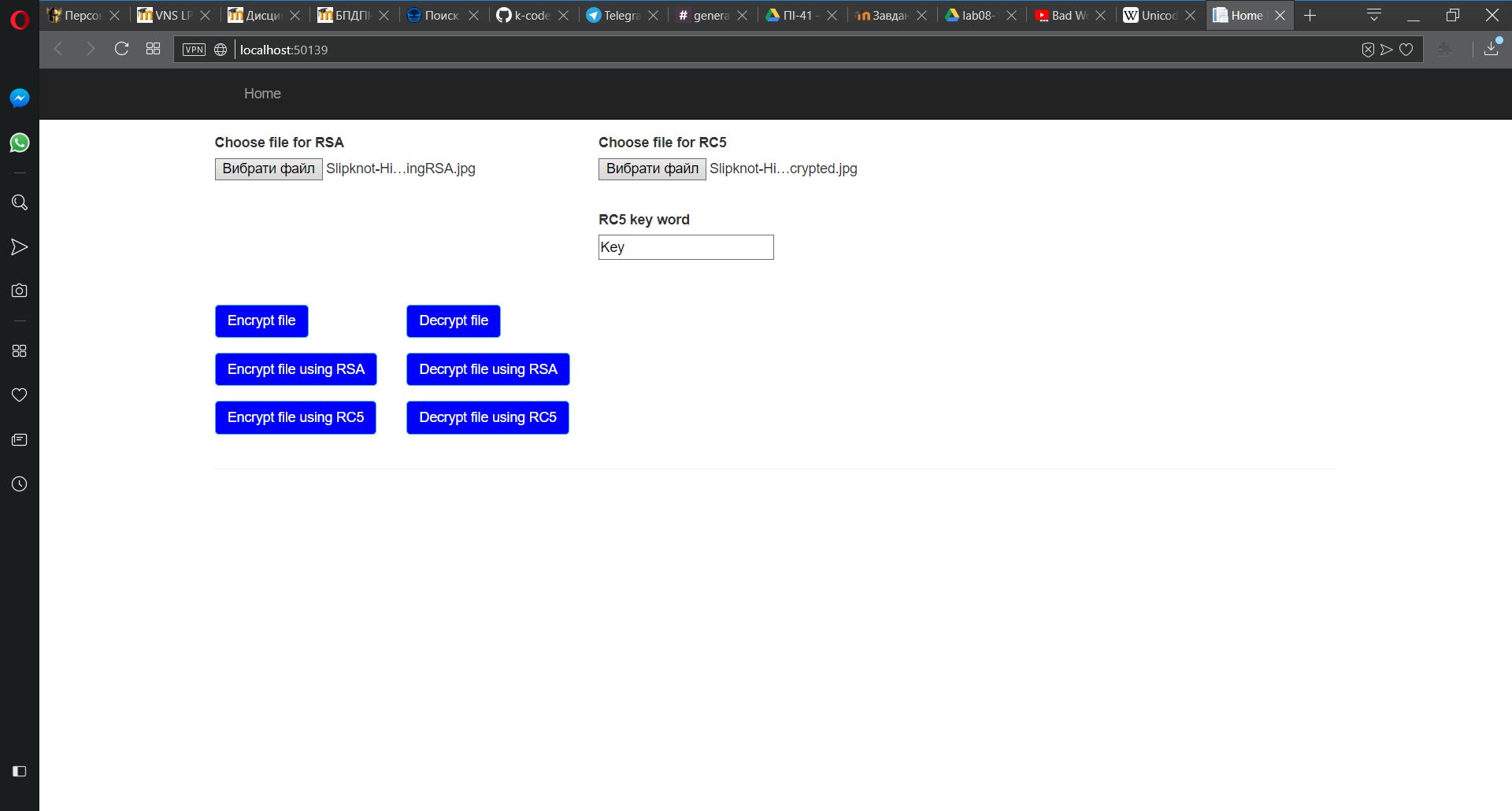


Рис. 1. Початковий інтерфейс застосунку

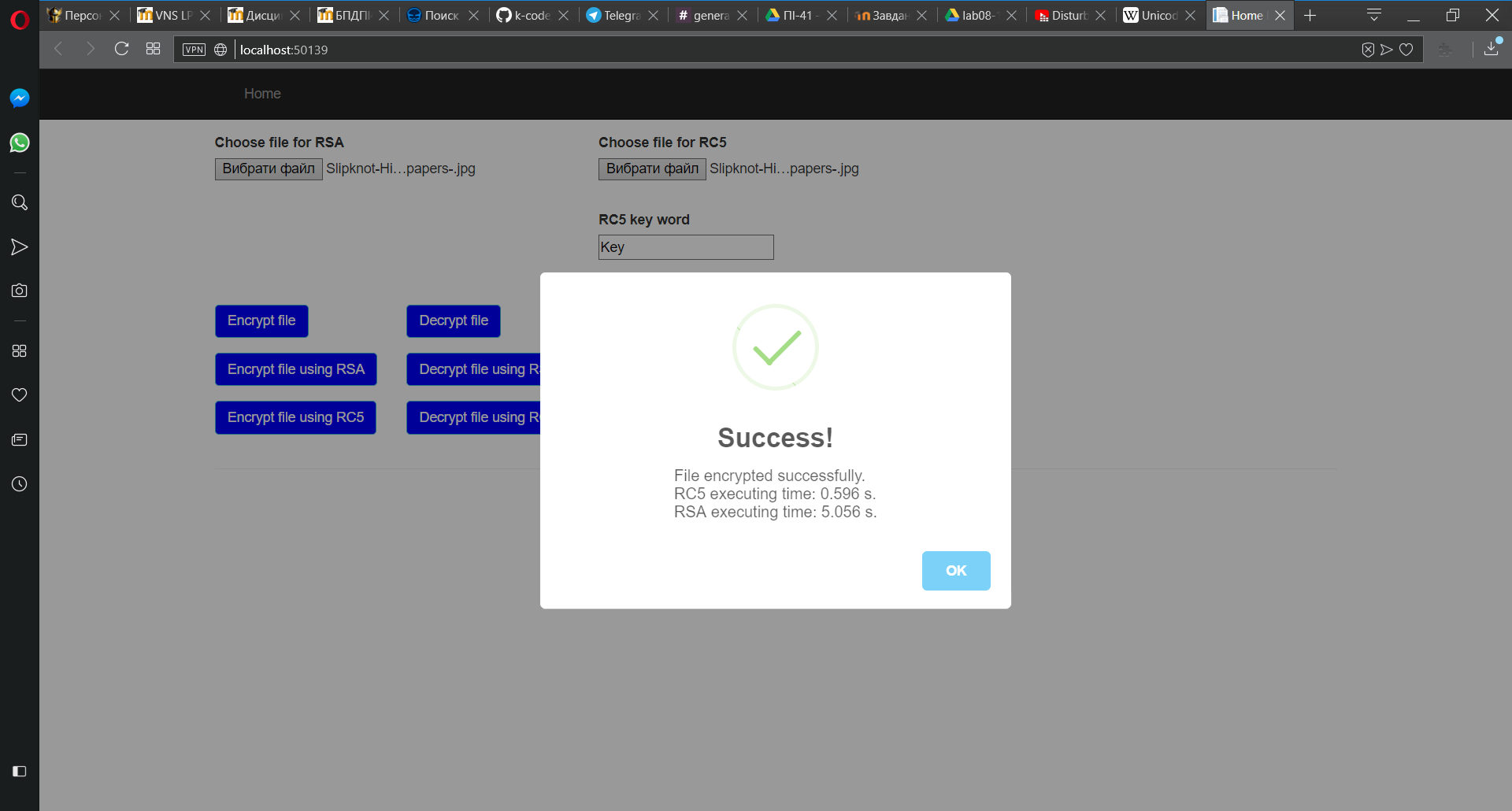


Рис. 2. Результати кодування файлу

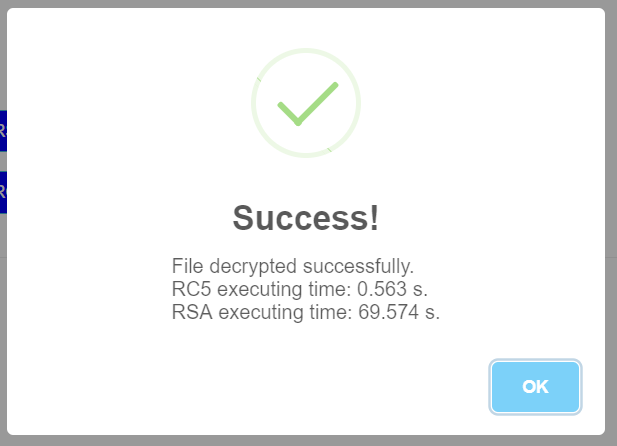


Рис. 3. Результати декодування файлу

Вихідний код програми

public class RSAService

{

public const string PublicKeyFile = "PublicKey";

public const string PrivateKeyFile = "PrivateKey";

public const int DefaultKeySize = 1024;

private const int EncryptBlockSize = 64;

private const int DecryptBlockSize = 128;

private RSACryptoServiceProvider Provider { get; set; } = new RSACryptoServiceProvider(DefaultKeySize);

private FileService FileService { get; set; } = new FileService();

public RSAKeysModel GenerateKeys(int keySize)

{

Provider = new RSACryptoServiceProvider(keySize);

var publicKey = Provider.ExportCspBlob(false);

var privateKey = Provider.ExportCspBlob(true);

return new RSAKeysModel { PublicKey = publicKey, PrivateKey = privateKey };

}

public byte[] Encrypt(byte[] input)

{

var publicKey = FileService.LoadFile(PublicKeyFile);

Provider.ImportCspBlob(publicKey);

var encryptedData = new MemoryStream();

for (int i = 0; i < input.Length; i += EncryptBlockSize)

{

var buffer = new byte[i + EncryptBlockSize > input.Length ? input.Length - i : EncryptBlockSize];

Array.Copy(input, i, buffer, 0, buffer.Length);

buffer = Provider.Encrypt(buffer, false);

encryptedData.Write(buffer, 0, buffer.Length);

}

return encryptedData.ToArray();

}

public byte[] Decrypt(byte[] input)

{

var privateKey = FileService.LoadFile(PrivateKeyFile);

Provider.ImportCspBlob(privateKey);

var decryptedData = new MemoryStream();

for (int i = 0; i < input.Length; i += DecryptBlockSize)

{

var buffer = new byte[i + DecryptBlockSize > input.Length ? input.Length - i : DecryptBlockSize];

Array.Copy(input, i, buffer, 0, buffer.Length);

buffer = Provider.Decrypt(buffer, false);

decryptedData.Write(buffer, 0, buffer.Length);

}

return decryptedData.ToArray();

}

}

public class HomeController : Controller

{

public RC5Service RC5Service { get; set; } = new RC5Service();

public FileService FileService { get; set; } = new FileService();

public RSAService RSAService { get; set; } = new RSAService();

public ActionResult Index()

{

var model = new InputVewModel { RC5Key = "Key" };

return View(model);

}

[HttpPost]

public JsonResult EncryptData(InputVewModel input)

{

try

{

Stopwatch stopwatch = new Stopwatch();

var inputRSAData = FileService.LoadFile(input.RSAFileInput);

var inputRC5Data = FileService.LoadFile(input.RC5FileInput);

stopwatch.Start();

var rc5EncryptingResults =

RC5Service.Encrypt(inputRC5Data, Encoding.Unicode.GetBytes(input.RC5Key), 64, 16, 32);

stopwatch.Stop();

var rc5Time = stopwatch.ElapsedMilliseconds / 1000.0;

FileService.SaveEncriptingResut(rc5EncryptingResults, input.RC5FileInput);

stopwatch.Reset();

var rsaKeys = RSAService.GenerateKeys(RSAService.DefaultKeySize);

FileService.SaveFile(RSAService.PublicKeyFile, rsaKeys.PublicKey);

FileService.SaveFile(RSAService.PrivateKeyFile, rsaKeys.PrivateKey);

stopwatch.Start();

var rsaEncryptingResults = RSAService.Encrypt(inputRSAData);

stopwatch.Stop();

var rsaTime = stopwatch.ElapsedMilliseconds / 1000.0;

FileService.SaveFileWithSuffix(input.RSAFileInput, "\_encryptedUsingRSA", rsaEncryptingResults);

var successMessage = $"File encrypted successfully. {Environment.NewLine} " +

$"RC5 executing time: {rc5Time} s.{Environment.NewLine} RSA executing time: {rsaTime} s.";

return Json(new { Success = true, SuccessMessage = successMessage }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult DecryptData(InputVewModel input)

{

try

{

Stopwatch stopwatch = new Stopwatch();

var inputRSAData = FileService.LoadFile(input.RSAFileInput);

var inputRC5Data = FileService.LoadFile(input.RC5FileInput);

stopwatch.Start();

var rc5DecryptingResults =

RC5Service.Decrypt(inputRC5Data, Encoding.Unicode.GetBytes(input.RC5Key), 64, 16, 32);

stopwatch.Stop();

var rc5Time = stopwatch.ElapsedMilliseconds / 1000.0;

FileService.SaveDecriptingResut(input.RC5FileInput, rc5DecryptingResults);

stopwatch.Reset();

RSAService.GenerateKeys(RSAService.DefaultKeySize);

stopwatch.Start();

var rsaDecryptingResults = RSAService.Decrypt(inputRSAData);

stopwatch.Stop();

var rsaTime = stopwatch.ElapsedMilliseconds / 1000.0;

FileService.SaveFileWithSuffix(input.RSAFileInput, "\_decryptedUsingRSA", rsaDecryptingResults);

var successMessage = $"File decrypted successfully. {Environment.NewLine} " +

$"RC5 executing time: {rc5Time} s.{Environment.NewLine} RSA executing time: {rsaTime} s.";

return Json(new { Success = true, SuccessMessage = successMessage }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult EncryptDataUsingRSA(InputVewModel input)

{

try

{

var rsaKeys = RSAService.GenerateKeys(RSAService.DefaultKeySize);

FileService.SaveFile(RSAService.PublicKeyFile, rsaKeys.PublicKey);

FileService.SaveFile(RSAService.PrivateKeyFile, rsaKeys.PrivateKey);

var encryptingResults = RSAService.Encrypt(FileService.LoadFile(input.RSAFileInput));

FileService.SaveFileWithSuffix(input.RSAFileInput, "\_encrypted", encryptingResults);

return Json(new { Success = true }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult DecryptDataUsingRSA(InputVewModel input)

{

try

{

var decryptingResults =

RSAService.Decrypt(FileService.LoadFile(input.RSAFileInput));

FileService.SaveFileWithSuffix(input.RSAFileInput, "\_decrypted", decryptingResults);

return Json(new { Success = true }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult EncryptDataUsingRC5(InputVewModel input)

{

try

{

var encryptingResults =

RC5Service.Encrypt(

FileService.LoadFile(input.RC5FileInput),

Encoding.Unicode.GetBytes(input.RC5Key), 64, 16, 32);

FileService.SaveEncriptingResut(encryptingResults, input.RC5FileInput);

return Json(new { Success = true }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

[HttpPost]

public JsonResult DecryptDataUsingRC5(InputVewModel input)

{

try

{

var decryptingResults =

RC5Service.Decrypt(

FileService.LoadFile(input.RC5FileInput),

Encoding.Unicode.GetBytes(input.RC5Key), 64, 16, 32);

FileService.SaveDecriptingResut(input.RC5FileInput, decryptingResults);

return Json(new { Success = true }, JsonRequestBehavior.AllowGet);

}

catch(Exception ex)

{

return Json(new { Success = false, ErrorMessage = ex.Message }, JsonRequestBehavior.AllowGet);

}

}

}

**Висновки**

На даній лабораторній роботі я ознайомився з методами і засобами криптографії з відкритим ключем, навчитись створювати програмні засоби з використанням криптографічних інтерфейсів. Як показали результати, навіть при роботі із файлами доволі невеликого розміру, алгоритм RSA суттєво поступається швидкодією алгоритму RC5, зокрема при декодуванні.