[**Лабораторна робота № 4**](http://vns.lp.edu.ua/mod/page/view.php?id=197094)

**СТВОРЕННЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМУ**[**ШИФРУВАННЯ**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary)**З ВІДКРИТИМ КЛЮЧЕМ**[**RSA**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)**З ВИКОРИСТАННЯМ MICROSOFT**[**CRYPTOAPI**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary)

**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](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) та ключа [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary), які неможливо обчислити один з одного. Перша праця, присвячені цій проблемі вийшла у 1976 році, і з того часу було створено багато алгоритмів, що використовують концепцію відкритих ключів. Загальна схема виглядає наступним чином:

1.     Кожен користувач генерує пару ключів: один для [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і один для [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary).

2.     Кожен користувач публікує свій ключ [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), 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](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) з відкритим ключем розроблялися для того, щоб вирішити дві найбільш важкі задачі, що виникли при використанні симетричного [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary).

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](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187776&displayformat=dictionary) з відкритим ключем дозволяє вирішити набагато ширше коло задач, ніж [криптографія](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187776&displayformat=dictionary) класична. Однак існує ряд причин, з яких асиметричні алгоритми [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) не можуть повноцінно замінити симетричні алгоритми:

        По-перше, алгоритми з таємним ключем набагато простіше реалізовуються як програмно, так і апаратно. Через це за однакових характеристик продуктивності та стійкості складність, а значить і ціна апаратних засобів, що реалізують шифр з відкритим ключем помітно вища за ціну апаратури, що реалізує класичний шифр, а при програмній реалізації на одному й тому ж типі процесора симетричні шифри працюють швидше асиметричних.

        По-друге, надійність алгоритмів з відкритим ключем на даний час обґрунтована набагато гірше, ніж надійність алгоритмів з таємним ключем і немає гарантії, що через деякий час вони не будуть розкриті, як це сталося з криптосистемою, заснованою на задачі про вкладання ранця.

Тому для організації шифрованого зв’язку на даний час застосовуються винятково класичні шифри, а методи криптографії з відкритим ключем використовуються тільки там, де вони не працюють, тобто для організації різних протоколів типу цифрового підпису, відкритого розподілу ключів тощо.

**Алгоритм**[**RSA**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)

[RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) (авторами є Rіvest, Shamіr і Adleman) – це алгоритм з відкритим ключем (асиметричний алгоритм), призначений як для [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), так і для аутентифікації. Був розроблений в 1977 році. З тих пір алгоритм [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) широко застосовується практично в усіх додатках, що використовують криптографію з відкритим ключем.

Алгоритм заснований на використанні того факту, що задача факторизації є складною, тобто легко перемножити два числа, у той час як не існує поліноміального алгоритму знаходження простих множників великого числа.

Алгоритм [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) є блоковим алгоритмом [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), де зашифровані і незашифровані дані є цілими числами в діапазоні між 0 і *n*-1 для деякого *n*.

Алгоритм, розроблений Райвестом, Шаміром і Адлеманом, використовує вирази з експонентами. Дані шифруються блоками, кожен блок розглядається як число, менше деякого числа *n*. [Шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary) мають такий вигляд для деякого незашифрованого блоку *M* і зашифрованого блоку *C*.

,

.

Як відправник, так і одержувач повинні знати значення *n*. Відправник знає значення *e*, одержувач знає значення *d*. Таким чином, відкритим ключем є *KU*={*e*,*n*}, а закритим ключем є *KR*={*d*,*n*}. При цьому повинні виконуватися наступні умови:

1.     Можливість знайти значення *e*, *d* та *n* такі, що  для усіх *M*<*n*.

2.     Відносна легкість обчислення *Me* і *Cd* для усіх значень *M*<*n*.

3.     Неможливість визначити *d*, знаючи *e* та *n*.

Тепер розглянемо сам алгоритм [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary). Нехай *p* та *q* – прості числа.

.

Таким чином, слід вибрати *e* і *d* такі, що  (тут *k* – ціле число,  – функція Ейлера). Це еквівалентно наступним співвідношенням:

,

.

Отже *e* і *d* є взаємно оберненими по множенню за модулем . Зауважимо, що це вірно тільки в тому випадку, якщо *d* (і отже, *e*) є взаємно простими з . Таким чином,  (gcd – найбільший спільний дільник). Усі елементи алгоритму наведені в табл. 5.

Таблиця 5. Елементи алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)

|  |  |
| --- | --- |
| **Елемент** | **Опис** |
| *p*, *q* – два простих цілих числа | закриті, вибираються |
|  | відкритий, обчислюється |
| *e*, таке що , | відкритий, вибирається |
|  | закритий, обчислюється |

Як [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), так і [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary) в [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) передбачають використання операції піднесення цілого числа до цілого степеня по модулю *n*. Якщо піднесення до степеня виконувати безпосередньо з цілими числами і потім проводити порівняння по модулю *n*, то проміжні значення виявляються дуже великими. Однак можна скористатись властивостями модульної арифметики: . Таким чином, можна розглядати проміжні результати по модулю *n*. Це робить обчислення практично виконуваними.

Як вже зазначалось асиметричні алгоритми [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і, зокрема, [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) є набагато повільнішими за симетричні алгоритми. Так, наприклад, апаратно [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) приблизно в 100–1000 разів повільніший за алгоритми симетричного [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary). На даний час розроблено мікросхеми, які, використовуючи 512-бітовий модуль, перейшли рубіж 1 Мбіт/с. Виробники також застосовують [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) в смарт-картах, але ці реалізації є повільнішими. Програмно алгоритми симетричного [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) приблизно у 10–100 разів швидші за [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary). Ці числа можуть незначно змінитись при зміні технології, однак [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) ніколи не досягне швидкості симетричних алгоритмів.

Швидкість програмної реалізації алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) можна суттєво збільшити, якщо правильно вибрати значення *e.* Трьома найбільш вживаними варіантами є 3, 17 та 65537 (216+1). (Двійкове представлення 65537 містить тільки дві одиниці, тому для піднесення до степеня треба виконати тільки 17 множень.) Стандарт Х.509 рекомендує використовувати значення 65537, РЕМ рекомендує 3, a PKCS#1 – 3 або 65537. Не існує жодних проблем безпеки, пов’язаних з використанням в якості *e* будь-якого з цих трьох значень (за умови доповнення повідомлення випадковими числами), навіть якщо одне й те саме значення *e* використовується цілою групою користувачів.

Операції із закритим ключем можна пришвидшити за допомогою китайської теореми про остачі, якщо зберегти значення *p* і *q*, а також додаткові значення: ,  і . Ці додаткові числа можна легко обчислити за закритим і відкритим ключами.

**Криптографічні інтерфейси та Microsoft**[**CryptoAPI**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary)

Підсистема безпеки комп'ютерної системи потребує реалізації ряду спільних функцій, пов'язаних з перетворенням вмісту об'єктів системи (файлів, записів бази даних) або з обчисленням деяких функцій зі спеціальними властивостями, які істотно залежать від вмісту об'єктів. До таких функцій відносяться алгоритми контролю цілісності об'єктів комп'ютерної системи, алгоритми аутентифікації або авторизації суб'єктів (чи користувачів, які управляють суб'єктами), алгоритми підтримки конфіденційності вмісту об'єктів (функції [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary)).

Суб'єкти (програми, процеси) комп'ютерної системи, пов'язані з виконанням захисних функцій, можуть використовувати деяку загальну підмножину функцій логічного перетворення об'єктів (зокрема, алгоритми [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і контролю цілісності об'єктів). При проектуванні і реалізації суб'єктів комп'ютерної системи, як правило, реалізується наступний підхід щодо використання спільного ресурсу – застосування розділювальних суб'єктів, що виконують функції, спільні для деякої підмножини зовнішніх по відношенню до даного суб'єкту функцій (динамічно завантажуваних бібліотек для Microsoft Windows). Логічно розповсюдити даний підхід на функції реалізації захисту. Серед множини функцій, що відносяться до захисної компоненти комп'ютерної системи, можна виділити три класи: функції, пов'язані з роботою засобів ідентифікації і авторизації користувачів, суб'єктів і об'єктів; функції, пов'язані з контролем незмінності об'єктів; функції, пов'язані з логічним перетворенням об'єктів системи і підтримкою функцій конфіденційності (криптографічні функції). Також окремо виділяють функції генерації випадкових послідовностей, необхідних, зокрема, для формування індивідуальних аутентифікаційних ознак або ключів користувачів.

Проблему проектування суб'єктів, що реалізовують функції логічного захисту, можна розглядати в декількох аспектах:

       оптимальна реалізація об'єктів в рамках деякого об'єкта комп'ютерної системи, до якого звертається решта суб'єктів за виконанням відповідних функцій (в даному випадку йдеться про завдання оптимізації параметрів "швидкодія – пам'ять" при реалізації захисних функцій);

       мобільність суб'єктів, що використовують захисні функції при зміні внутрішнього наповнення, яке реалізує захисні функції об'єкта (мається на увазі задача максимальної переносимості суб'єктів, що використовують захисні функції на інші алгоритми, наприклад на інший алгоритм [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary));

       коректне використання об'єкта, що реалізовує захисні функції, з боку сторонніх модулів, що його викликають.

Технологія застосування розділювальних функцій логічного захисту – це порядок використання засобів логічного захисту інформації в системі, при якому:

       не вимагається змін в програмному забезпеченні (у змісті і складі суб'єктів комп'ютерної системи) при зміні алгоритмів захисту;

       у комп'ютерній системі однозначно виділяється модуль реалізації захисних функцій.

Відкритим інтерфейсом модуля реалізації захисних функцій називають детальну специфікацію функцій, реалізованих в модулі, що дозволяє організувати виконання цих функцій із зовнішніх суб'єктів.

"Відкритість" інтерфейсу розуміється як його повний опис для використання зовнішніми суб'єктами реалізованих в модулі функцій. Для проектування схем інформаційних потоків комп'ютерної системи, зокрема тих, що мають відношення до забезпечення безпеки, такий опис виконує ключову роль.

В даний час існує досить багато криптографічних інтерфейсів. Однак розглянемо тільки базові криптографічні перетворення, тобто криптографічні інтерфейси, що в явному вигляді реалізовують основні криптографічні операції: [симетричне шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187819&displayformat=dictionary), хешування, цифровий підпис і несиметричний обмін ключами. Одним з найбільш поширених є Microsoft [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) 2.0. Розповсюдження [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) пов'язане не тільки з його зручністю, документованістю та іншими об'єктивними чинниками. Найважливішим чинником є те, що Microsoft найактивнішим чином інтегрувала його в свої операційні системи і прикладні програми. Сучасні операційні системи Microsoft (Windows 2000, Windows XP, Windows Vista) містять багато криптографічних підсистем різного призначення як прикладного рівня, так і рівня ядра, і ключову роль в реалізації цих підсистем відіграє інтерфейс [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary). Зрозуміло, в підсистемах рівня ядра базові криптографічні перетворення здійснюються безпосередньо в драйверах, що реалізовують ці підсистеми. Функції [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) в таких випадках використовуються для допоміжних операцій на прикладному рівні.

Далі детальніше розглянемо роботу саме інтерфейсу [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary), точніше, набору базових криптографічних функцій (base cryptography functions), який називають також інтерфейсом [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) 1.0.

Інтерфейс Microsoft [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) 2.0 містить як функції, що здійснюють базові криптографічні перетворення, так і функції, що реалізовують перетворення вищого рівня – роботу з сертифікатами Х.509, роботу з криптографічними повідомленнями PKCS#7 та інші функції, що підтримують так звану інфраструктуру відкритих ключів (Public Key Infrastructure).

На основі використання можливостей базових криптографічних функцій можна не тільки більш глибоко розуміти роботу всього інтерфейсу [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) 2.0, але й створювати власні криптографічні підсистеми будь-якого рівня.

Реалізація всіх алгоритмів ([шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), цифрового підпису і т.п.) повністю винесена із складу самого [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) і реалізується в окремих незалежних динамічних бібліотеках – "[криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)ах" (Cryptographic Service Provider – CSP).

[Криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)ом називають незалежний модуль, що забезпечує безпосередню роботу з криптографічними алгоритмами. Кожен [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary) повинен забезпечувати:

        реалізацію стандартного інтерфейсу [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а;

        роботу з ключами [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), призначеними для забезпечення роботи алгоритмів, специфічних для даного [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а;

        неможливість втручання третіх осіб в схему роботи алгоритмів.

[Криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)и реалізуються у вигляді динамічно завантажуваних бібліотек (DLL). Таким чином, достатньо важко втрутитись у хід виконання алгоритму, реалізованого в [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)і, оскільки компоненти криптосистеми Windows повинні мати цифровий підпис. У [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)ів повинні бути відсутні можливості зміни алгоритму через встановлення його параметрів. Таким чином вирішується задача забезпечення цілісності алгоритмів [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а.

[CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) надає наступні стандартні [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)и:

        Microsoft Base Cryptographic Provider

        Microsoft Strong Cryptographic Provider

        Microsoft Enhanced Cryptographic Provider

        Microsoft AES Cryptographic Provider

        Microsoft [DSS](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187738&displayformat=dictionary) Cryptographic Provider

        Microsoft Base [DSS](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187738&displayformat=dictionary) and Diffie-Hellman Cryptographic Provider

        Microsoft [DSS](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187738&displayformat=dictionary) and Diffie-Hellman/Schannel Cryptographic Provider

        Microsoft [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)/Schannel Cryptographic Provider

Уся архітектура CryptoAPI може бути розділена три основні частини:

        Базові функції;

        Функції [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і роботи з сертифікатами;

        Функції роботи з повідомленнями.

***Базові функції***

До цієї групи входять функції для вибору і підключення до [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а, генерації і зберігання ключів, обміну ключами. Сюди так само входять можливості управління параметрами ключа, такими як: режим зчеплення блоків, вектор ініціалізації.

До базових функцій також можна віднести криптографічну функцію генерації випадкових даних. Це найбільш важлива функція у всій криптосистемі, оскільки вона користується для генерації ключів. Вся відповідальність за її реалізацію лягає на розробників [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а.

Основні функції в цій групі: *CryptAcquireContext, CryptReleaseContext, CryptGenKey, Crypt*[*Des*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187737&displayformat=dictionary)*troyKey, CryptExportKey, CryptImportKey, CryptDeriveKey, CryptGenRandom.*

***Функції***[***шифрування***](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary)***і для роботи з сертифікатами***

До цієї групи входять всі функції для хешування даних, [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary), а також функції для використання сертифікатів, основним завданням яких є надання доступу до відкритого ключа. [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) підтримує [сертифікати](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187816&displayformat=dictionary) специфікації X.509, в які входить інформація про версію сертифіката, його серійний номер, періоді дії, алгоритм [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) публічного ключа та ін.

Основні функції в цій групі: *CryptEncrypt, CryptDecrypt, CryptCreateHash, Crypt*[*Des*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187737&displayformat=dictionary)*troyHash, CryptHashData, CryptSignHash, CertOpenStore, CertCloseStore, CertFindCertificateInStore.*

***Функції для роботи з повідомленнями***

Під повідомленнями в [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) розуміються дані в стандартизованому форматі PKCS #7, розробленому [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) Laboratories. Функції для роботи з ними діляться на дві частини: високорівневі функції (спрощені) і низькорівневі.

**Використання функцій**[**CryptoAPI**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary)**для реалізації алгоритмів**[**шифрування**](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary)

Будь-який сеанс роботи з [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) починається з ініціалізації (отримання контексту)[[1]](http://vns.lp.edu.ua/mod/page/view.php?id=189886" \l "_ftn1" \o "). Ініціалізація здійснюється за допомогою функції CryptAcquireContext. В якості параметрів ця функція приймає ім’я контейнеру ключів, ім’я [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а, тип провайдера і «прапорці», що визначають тип і дії з контейнером ключів і режим роботи [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary)а.

*HCRYPTPROV hProv;*

*::CryptAcquireContext(&hProv,/\*параметр\*/,ms\_enhanced\_prov,prov\_*[*rsa*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*\_full,0)*

[Криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary) – це бібліотека, що реалізовує певний набір криптографічних алгоритмів і яка забезпечує роботу з ними. Існує близько семи стандартних провайдерів, встановлених в системі. Так, наприклад, Microsoft Base Cryptographic Provider вказується як MS\_DEF\_PROV, а Microsoft Enhanced Cryptographic Provider – як MS\_ENHANCED\_PROV.

Кожен [криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary) відноситься до певного типа. Це дозволяє, перебравши всі встановлені на машині провайдери, вибрати ті, які підтримують потрібні алгоритми. Два згадані провайдери мають тип PROV\_[RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)\_FULL.

[Криптопровайдер](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187781&displayformat=dictionary) підтримує захищені області, звані контейнерами ключів. Контейнери дозволяють додаткам зберігати і використовувати згенеровані ключі, забезпечуючи захист самого ключа.

Контейнери бувають двох типів – користувацькі (цей тип використовується за умовчанням) і машинні (CRYPT\_[MAC](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187742&displayformat=dictionary)HINE\_KEYSET). Користувацький контейнер доступний тільки додаткам, що виконуються від імені власника контейнера. Додаток може використовувати такий контейнер для збереження персональних ключів. Доступ до машинних контейнерів дозволений тільки адміністраторам. У них зазвичай зберігаються ключі, що використовуються службами і системними програмами. Тип контейнера задається прапорцем при отриманні контексту.

Для первинного створення контейнера потрібно викликати CryptAcquireContext з прапорцем CRYPT\_NEWKEYSET. Для видалення контейнера потрібно вказати прапор CRYPT\_DELETEKEYSET.

Якщо додатку не потрібен доступ до контейнера ключів (наприклад, додаток обчислює хеш [MD5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187743&displayformat=dictionary)), то варто викликати CryptAcquireContext з прапорцем CRYPT\_VERIFYCONTEXT, передаючи NULL замість імені контейнера.

Деініціалізація [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) виконується за допомогою функції CryptReleaseContext, єдиним значущим параметром якої є одержаний раніше дескриптор криптографічного контексту.

Для генерації ключів в [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) передбачені дві функції – CryptGenKey і CryptDeriveKey. Перша з них генерує ключі випадковим чином, а друга – на основі даних користувача. При цьому гарантується, що для одних і тих же вхідних даних CryptDeriveKey завжди видає один і той же результат. Цей спосіб генерації ключів може бути корисним для створення симетричного ключа [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) на базі пароля. Зупинимося детальніше на функції CryptGenKey, яка використовується найчастіше. Ця функція має прототип:

*BOOL WINAPI CryptGenKey(HCRYPTPROV hProv, ALG\_ID Algid, DWORD dwFlags, HCRYPTKEY\* phKey);*

Перший і четвертий параметри говорять самі за себе. Другим параметром передається ідентифікатор алгоритму [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary), для якого генерується ключ (наприклад CALG\_3[DES](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187737&displayformat=dictionary)). При генерації ключових пар [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) для [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) і підпису використовуються спеціальні значення AT\_KEYEXCHANGE і AT\_SIGNATURE. Третій параметр задає різні опції ключа, які залежать від алгоритму і провайдера. Наприклад, старші 16 бітів цього параметра можуть задавати розмір ключа для алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary). Докладний опис всіх прапорів можна знайти в MSDN. Параметри, які не можна задати при генерації ключа (наприклад, вектори, ініціалізацій), можна встановити вже після його створення за допомогою функції CryptSetKeyParam.

Обмін ключами в [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) реалізується за допомогою функцій CryptExportKey і CryptImportKey, що мають наступні прототипи:

*BOOL WINAPI CryptExportKey(HCRYPTKEY hKey,HCRYPTKEY hExpKey,DWORD dwBlobType,DWORD dwFlags,BYTE\* pbData,DWORD\* pdwDataLen);*

*BOOL WINAPI CryptImportKey(HCRYPTPROV hProv,BYTE\* pbData,DWORD dwDataLen,HCRYPTKEY hImpKey,DWORD dwFlags,HCRYPTKEY\* phKey);*

В якості ключів експорту/імпорту можуть використовуватися або ключова пара [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) (з типом AT\_KEYEXCHANGE), або симетричний [сеансовий ключ](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187814&displayformat=dictionary). Параметр dwBlobType залежить від того, який ключ експортується (імпортується), і задає тип структури, в яку поміщається ключ, що експортується. Для відкритого ключа це PUBLICKEYBLOB, і ключ експорту/імпорту при цьому позбавлений сенсу і повинен бути нулем. Для закритого ключа це PRIVATEKEYBLOB, і як ключ експорту може використовуватися [сеансовий ключ](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187814&displayformat=dictionary). Для сеансового ключа це зазвичай SIMPLEBLOB, а експортується він, як правило, на відкритому ключі одержувача.

Параметри pbData і pdwDataLen задають адресу і розмір буфера під структуру ключа, що експортується. Якщо розмір структури не відомий при написанні програми, то можна визначити необхідний розмір буфера, встановивши в pbData нуль. В цьому випадку потрібний розмір повертається в pdwDataLen.

Після закінчення роботи з ключем, його потрібно знищити викликом Crypt[Des](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187737&displayformat=dictionary)troyKey.

Наступний приклад демонструє створення і експорт пари ключів для [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary):

*//створення і експорт пари ключів*[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)

*if(CryptGenKey(hProv,AT\_KEYEXCHANGE,1024<<16,&hKey)) //генеруємо 1024-бітовий ключ*

*{*

[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PubKey1024 key;*

*DWORD dwLen=sizeof(*[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PubKey1024);*

*if(CryptExportKey(hKey,null,publickeyblob,0,(byte \*)&key,pdwLen)) //експорт ключа*

*{*

*//... виконуються певні дії*

*}*

*Crypt*[*Des*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187737&displayformat=dictionary)*troyKey(hKey); //знищуємо ключ*

*}*

Структура типу [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)PubKey1024 має наступний вигляд:

*template<int bitlen> struct*[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PubKey*

*{*

*PUBLICKEYSTRUC  publickeystruc ;*

[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PUBKEY*[*rsa*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*pubkey;*

*BYTE modulus[bitlen/8];*

*};*

*template<int bitlen> struct*[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PrivKey*

*{*

[*RSA*](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)*PubKey<bitlen> pubkey;*

*BYTE prime1[bitlen/16];*

*BYTE prime2[bitlen/16];*

*BYTE exponent1[bitlen/16];*

*BYTE exponent2[bitlen/16];*

*BYTE coefficient[bitlen/16];*

*BYTE privateExponent[bitlen/8];*

*};*

Починаючи з Windows 2000 розширений (Enhanced) кріптопровайдер підтримує безпосереднє [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) даних за алгоритмом [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary). Максимальний розмір даних, які можна зашифрувати за один виклик функції Cryptencrypt, дорівнює розміру ключа мінус 11 байт. Справа в тому, що при шифруванні додається обов'язковий заповнювач (padding), який згодом перевіряється при дешифруванні. Відповідно, використання шифру [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) може бути доцільним тільки при невеликих об'ємах шифрованих даних (наприклад, при обміні ключами) через істотне збільшення об'єму шифрованого тексту і повільну роботу алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) в порівнянні з блоковими шифрами.

Для [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) даних [відкритий текст](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187760&displayformat=dictionary) зчитується блоками із вхідного файлу, а результат [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) записується у вихідний файл, [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) здійснюється функцією:

*::CryptEncrypt(hKey,hHash,fSize<=BUFFER\_SIZE,0,buf,&dwSzLow,sizeof(buf)),*

а [дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary):

*::CryptDecrypt(hKey,hHash,fEncSize<=BUFFER\_SIZE,0,buf,&dwSzLow))*

Параметр hHash дозволяє паралельно з [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary)м/[дешифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187765&displayformat=dictionary)м здійснювати хешування даних для подальшого електронного підпису або його перевірки. Прапор Final визначає, чи є шифрований блок даних останнім. Він необхідний, оскільки дані можна шифрувати по шматках, але для останнього блоку завжди виконується певна деініціалізація алгоритму (звільняються внутрішні структури), і багато алгоритмів проводять додавання (і перевірку коректності при дешифруванні) заповнювача (padding) після основних даних. Параметри pbData і pdwDataLen задають адресу буфера і розмір шифрованих даних. Для не останнього блоку даних (Final=FALSE) розмір даних повинен бути завжди кратний розміру шифрованого алгоритмом блоку. Для останнього блоку допускається порушення цієї умови.

Приклад [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) наведено в наступному коді:

*BYTE buf[BUFFER\_SIZE+8]; //8 – запас на padding*

*while(fSize)*

*{*

*if(!::ReadFile(hInFile,buf,BUFFER\_SIZE,pdwLen,NULL)) //читаємо блок даних*

*break;*

*dwSzLow=dwLen;*

*if(!::CryptEncrypt(hKey,hHash,fSize<=BUFFER\_SIZE,0,buf,&dwSzLow,sizeof(buf))) //шифруємо і хешуємо його*

*break;*

*if(!::WriteFile(hOutFile,buf,&dwSzLow,pdwSzLow,NULL))*

*break;*

*fSize-=dwLen;*

*}*

**Завдання до виконання роботи**

З використання функцій [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) створити програмну реалізацію алгоритму [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary). Оцінити швидкість [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary) та порівняти її зі швидкістю [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) алгоритму [RC5](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187748&displayformat=dictionary), реалізованого в роботі № 3, зробити відповідні висновки та відобразити їх у звіті до лабораторної роботи.

**Контрольні запитання.**

1.     Опишіть схематично використання алгоритмів [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary) з відкритим ключем.

2.     Для яких задач створені алгоритми з відкритим ключем?

3.     Основні недоліки алгоритмів з відкритим ключем.

4.     Області використання алгоритмів симетричного і асиметричного [шифрування](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187834&displayformat=dictionary).

5.     На якій складній задачі засновано алгоритм [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)?

6.     Як можна збільшити швидкодію програмної реалізації алгоритму [RSA](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187747&displayformat=dictionary)?

7.     Що включають в інтерфейс [CryptoAPI](http://vns.lp.edu.ua/mod/glossary/showentry.php?courseid=8932&eid=187736&displayformat=dictionary) 1.0?