**Лабораторна робота №7.**
**Використання CryptoAPI операційної системи Windows XP для створення криптографічного ПЗ.**

## 1.1 Мета роботи

*ознайомити студентів з методами роботи з CryptoAPI ОС Windows 2000/XP.*

## 1.2 Теоретичні відомості

Правильне функціонування підсистеми безпеки комп‘ютерної системи вимагає реалізації ряду функцій загального призначення, пов‘язаних з перетворенням вмісту об‘єктів системи (файлів, записів бази даних тощо) або з обчислення деяких спеціальних функцій, які суттєво залежать від вмісту об‘єктів. До таких функцій належать алгоритми контролю цілісності об‘єктів, аутентифікації та авторизації об‘єктів, що керують процесами, а також алгоритми підтримання конфіденційності інформації, що міститься в об‘єктах комп‘ютерної системи.

Міжнародні та національні стандарти описують ряд добре відомих та вивчених функцій захисного характеру, зокрема алгоритми хешування MD5, MD2, SHA тощо; алгоритми генерування та перевірки електронного цифрового підпису RSA, DSS та інших. Усі ці алгоритми мають різні механізми викликів (зокрема, різну довжину аргументів). Це, у свою чергу, означає, що вони несумісні між собою.

Тому задача вбудовування тих чи інших захисних механізмів в операційну систему на основі якогось одного алгоритму буде виглядати неефективною, особливо, якщо ця ОС розповсюджується в різних регіонах земної кулі. В цьому випадку логічним є побудова «шаруватої» структури, де окремий шар, реалізований, скажемо, як набір динамічних бібліотек, відповідає за захист інформації. Цей спосіб досить універсальний і широко застосовується у сімействі операційних систем Windows. Таким способом можна розв‘язати великий клас задач, пов‘язаних з універсалізацією ОС: від національних налаштувань системи до реалізації різноманітних засобів безпеки.

Зрозуміло, що такі структури повинні мати т.зв. «відкритий інтерфейс», тобто бути детально документованими для того, щоби програмісти могли використати засоби цієї структури при створенні прикладного програмного забезпечення, в тому числі і для захисту інформації.

Сьогодні є достатня кількість криптографічних інтерфейсів, однак найбільшої популярності набув інтерфейс від Microsoft - Microsoft CryptoAPI. Зараз використовується CryptoAPI версії 2.0. Причина популярності цього інтерфейсу полягає в тому, що Microsoft інтенсивно впровадила захисні механізми CryptoAPI у свої операційні системи та прикладне програмне забезпечення. Сучасні ОС сімейства Windows містять багато криптографічних підсистем різного призначення як прикладного рівня, так і

рівня ядра. Провідну роль в цьому грають якраз функції CryptoAPI, зокрема базові криптографічні функції, сукупність яких створює інтерфейс CryptoAPI 1.0.

Інтерфейс CryptoAPI 2.0 містить як базові криптографічні функції, так і функції, що реалізують перетворення вищого рівня – роботу з сертифікатами X.509, обробку криптографічних повідомлень PKCS#7 та інші функції, що підтримують інфраструктуру відкритих ключів. Однак набір базових криптографічних функцій цього інтерфейсу утворює CryptoAPI 1.0. Таким чином, функції CryptoAPI 1.0 утворюють криптографічне ядро прикладного рівня для сучасних операційних систем лінійки Windows.

Загальну архітектуру CryptoAPI 1.0 подано на рис. 1.

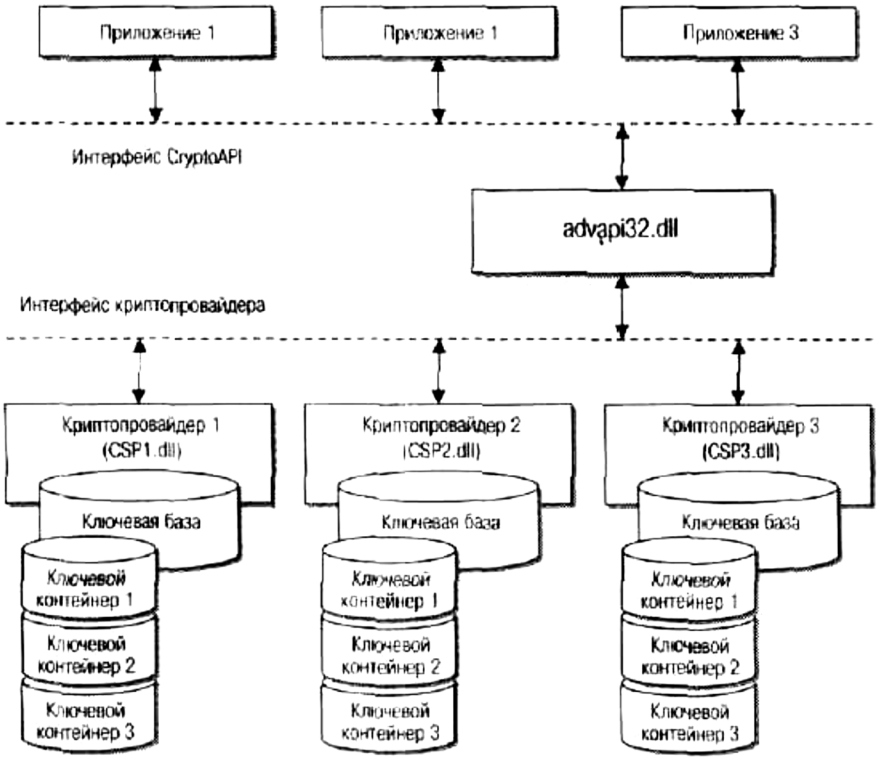


Рис. 1. Загальна архітектура CryptoAPI 1.0.

Усі функції інтерфейсу зосереджено у бібліотеці advapi32.dll. Ці процедури виконують ряд допоміжних функції та викликають бібліотеки, де безпосередньо реалізовано відповідні криптографічні перетворення. Такі бібліотеки називають ***криптопровайдерами***. Криптопровайдери мають стандартний набір функцій, який налічує 23 обов‘язкових та 2 необов‘язкових процедури. Ці процедури подано у таблиці 1.

Таблиця 1

Функції криптопровайдерів

| **Функція** | **Короткий опис** |
|---|---|
| *CryptAcquireContext* | Використовується для створення дескриптора ключового контейнера у рамках визначеного криптопровайдера (КП). |
| *CryptContextAddRef* | Збільшує на одиницю лічильник посилань на дескриптор КП. |
| *CryptEnumProviders* | Використовується для отримання першого та наступних доступних КП. |
| *CryptEnumProviderTypes* | Використовується для отримання першого та наступних доступних типів КП. |
| *CryptGetDefaultProvider* | Повертає КП, встановлений за замовчуванням для вказаного типу КП. |
| *CryptGetProvParam* | Повертає параметри КП |
| *CryptReleaseContext* | Вивільняє дескриптор КП |
| *CryptSetProvider*<br>*CryptSetProviderEx* | Задає тип та назву КП за замовчуванням |
| *CryptSetProvParam* | Встановлює параметри КП |
| *CryptDeriveKey* | Створює сесійні криптографічні ключі з ключового матеріалу |
| *CryptDestroyKey* | Звільняє дескриптор ключа |
| *CryptDuplicateKey* | Робить копію криптографічного ключа |
| *CryptExportKey* | Експортує криптографічні ключі із заданого контейнера |
| *CryptGenKey* | Генерує випадкові криптографічні ключі та ключові пари |
| *CryptGenRandom* | Генерує випадкову послідовність та зберігає її в буфері |
| *CryptGetKeyParam* | Повертає параметри ключа |
| *CryptImportKey* | Імпортує криптографічні ключі з ключового блоба у контейнер КП |
| *CryptSetKeyParam* | Встановлює параметри ключа |
| *CryptDecrypt* | Виконує операцію розшифрування даних |
| *CryptEncrypt* | Виконує операцію за шифрування даних |
| *CryptCreateHash* | Створює хешований потік даних |
| *CryptDestroyHash* | Знищує об‘єкт хеш функції |
| *CryptDuplicateHash* | Створює точну копію хеш-об‘єкта |

| | |
|---|---|
| *CryptGetHashParam* | Повертає параметри хеш-об‘єкта |
| *CryptHashData* | Додає дані до хеш-об‘єкта |
| *CryptHashSessionKey* | Підмішує до хеш-об‘єкта сесійний ключ |
| *CryptSetHashParam* | Встановлює параметри хеш-об‘єкту |
| *CryptSignHash* | Обчислює значення ЕЦП від значення хешу |
| *CryptVerifySignature* | Перевіряє ЕЦП заданого значення хешу |

Як бачимо з таблиці, CryptoAPI 1.0 підтримує усі основні методи криптографічного перетворення даних: від генерування криптографічних послідовностей випадкових чисел до операцій з електронним цифровим підписом. Таким чином, знаючи інтерфейс CryptoAPI 1.0, програміст може досить легко реалізувати усі популярні криптографічні алгоритми у своїх прикладних програмах.

Програміст, який працює з цим інтерфейсом, може отримати усю необхідну інформацію про певного криптопровайдера засобами функції *CryptGetProvParam*. Перше, що необхідно знати при цьому – це набір криптографічних стандартів, які реалізують встановлені у системі крипто-провайдери.

Окрім різниці у стандартах, криптопровайдери відрізняються способом фізичної організації збереження ключової інформації. З точки зору програмування спосіб зберігання ключів значення не має, однак він дуже важливий з точки зору експлуатації та безпеки комп‘ютерної системи. Існуючі криптопровайдери Microsoft зберігають ключову інформацію на жорсткому диску (у реєстрі або у файлах), а провайдери інших фірм (GemPlus, Schlumberger та Infineon) – на смарт-картках.

Якщо способи фізичної організації збереження ключової інформації у криптопровайдерів відрізняється, то логічна структура, яка визначається інтерфейсами та з якою мають справу програмісти, однакова для будь-якого типу провайдера. Ключова база визначається набором ключових контейнерів, кожен з яких має ім‘я, що привласнюється йому при створенні, а потім використовується для роботи з ним. У ключовому контейнері зберігається довготривала ключова інформація, наприклад, ключові пари для цифрового підпису або несиметричної системи шифрування.

Тепер розглянемо детально, як функції інтерфейсу CryptoAPI викликають бібліотеки конкретного криптопровайдера. Кожен криптопровайдер має своє власне ім‘я та тип. Його ім‘я – просто рядок, за допомогою якого система його ідентифікує. Так, базовий криптопровайдер Microsoft має назву Microsoft Base Cryptographic Provider v1.0. Тип криптопровайдера – ціле число (у нотації C – DWORD), значення якого ідентифікує набір криптографічних алгоритмів, що підтримуються. Криптопровайдер Microsoft має тип 1, цей тип провайдера реалізує в якості алгоритмів цифрового підпису та обміну ключів алгоритм RSA. Інший

базовий криптопровайдер Microsoft, „Microsoft Base DSS and Diffie-Hellman Cryptographic Provider", має тип 13. Цей тип криптопровайдера реалізує алгоритм цифрового підпису DSS, а в якості алгоритму обміну ключами – протокол Діффі-Хелмана.

Отже, для роботи з набором криптопровайдерами у системному реєстрі міститься список імен усіх криптопровайдерів. З кожним ім'ям пов'язаний тип криптопровайдера та ім'я бібліотеки, яка реалізує його алгоритми.
Окрім цього в системі міститься інформація про те, який криптопровайдер треба застосовувати, якщо користувач явно не вказав конкретне його ім'я, лише визначивши тип провайдера. Такий криптопровайдер називають провайдером за замовчуванням для заданого типу. Наприклад, для типу 1 провайдером за замовчуванням є Microsoft Base Cryptographic Provider v1.0, а для типу 13 - Microsoft Base DSS and Diffie-Hellman Cryptographic Provider. Для визначення криптопровайдерів за замовчуванням використовують функцію *CryptGetDefaultProvider,* а для зміни цього параметру – функції *CryptSetProvider* або *CryptSetProviderEx*. Функції дозволяють встановити провайдера за замовчуванням як для поточного користувача, так і для системи в цілому (усіх користувачів). Ці параметри зберігаються у вулику реєстру HKEY_LOCAL_MACHINE. Параметри, встановлені для поточного користувача, мають пріоритет над параметрами, встановленими для усієї системи, та зберігаються у вулику реєстру HKEY_CURRENT_USER. Якщо параметри для поточного користувача відсутні, застосовуються загальносистемні.

Тепер розглянемо, яким чином користувач починає працювати з конкретним криптопровайдером, і як система викликає конкретну бібліотеку, що відповідає обраному криптопровайдеру.

Робота з певним провайдером починається з виклику функції *CryptAcquireContext*, де користувач визначає тип потрібного криптопровайдера, його назву та назву робочого ключового контейнера. В результаті роботи функція повертає користувачу дескриптор криптопровайдера (handle), за допомогою якого користувач в подальшому буде звертатися до нього та передавати його у процедури для виконання усіх необхідних криптографічних операцій.

Детальний опис контексту роботи з криптопровайдерами та приклади (мовою програмування *C*) дивіться у книжці Щербакова Л.Ю., Домашева А.В. «Прикладная криптография».

Власне бібліотеки CryptoAPI разом з файлами заголовків та допомоги постачаються у складі бібліотек MSDN.

### *Практична частина*

Студентам пропонується скласти власну програму з використанням функцій CryptoAPI 1.0. Кожен варіант повинен являти собою закінчений програмний продукт, що вирішує одну з важливих сучасних задач криптографії.

Мова програмування значення не має.

Перелік завдань подано в таблиці 2. Розподіл завдань між студентами виконує викладач, що проводить лабораторні заняття.

Результатом роботи студента є програмний продукт, що працює без помилок та виконує усі функції, які вимагаються від нього (див. табл.2).

По результатах виконання лабораторної роботи студенти складають звіт, у якому детально описують правила роботи з розробленою програмою та наводять контрольні приклади для кожної її функціональної можливості.

Таблиця 2

Вимоги до варіантів завдань лабораторної роботи №10.

| **Варіант** | **Функціональні вимоги** |
|---|---|
| 1. Система аутентифікації на основі шифрування випадкових чисел | 1. Пароль, що його уводять при реєстрації користувача в системі, запам‘ятовується у базі даних системи та прив‘язується до певного ідентифікатора особи – логіна.<br>2. При спробі аутентифікації користувач уводить логін та пароль з клавіатури, користуючись клієнтською частиною програми.<br>3. Клієнтська частина генерує випадкове число, шифрує його одним з симетричних алгоритмів та відправляє логін, випадкове число і результати шифрування серверній частині.<br>4. Серверна частина вибирає з бази даних пароль, що відповідає отриманому логіну, шифрує тим самим алгоритмом отримане випадкове число на паролі з бази даних та порівнює результат з отриманим від клієнтської частини.<br>5. В залежності від результату порівняння зашифрованих чисел система робить висновок про те, допускати користувача до ресурсів системи, чи ні. Висновок серверної частини передається клієнтській, яка й повідомляє про це користувача.<br>6. Необхідно передбачити неуспішне завершення аутентифікації в разі триразового уведення неправильного пароля.<br>Перевагою такої системи аутентифікації є те, що мережею передається лише випадкове число та результат його зашифрування. Це підвищує стійкість системи аутентифікації по відношенню до перехоплення мережного трафіку. |

| | |
|---|---|
| | Алгоритм шифрування та довжину ключа, а також спосіб перетворення паролю у ключ шифрування розробіть самостійно. |
| 2.Система аутентифікації на основі хешування парольної інформації | 1. Пароль, що його уводять при реєстрації користувача в системі, хешується певним алгоритмом, до хеш-образу підмішується логін, отриманий образ запам‘ятовується у базі даних системи та прив‘язується до логіна.<br>2. При спробі аутентифікації користувач уводить логін та пароль з клавіатури, користуючись клієнтською частиною програми.<br>3. Клієнтська частина обчислює хеш-образ пароля та відправляє логін, і хеш-образ серверній частині.<br>4. Серверна частина вибирає з бази даних хеш-образ, що відповідає отриманому логіну, та порівнює його з отриманим образом.<br>5. В залежності від результату порівняння система робить висновок про те, допускати користувача до ресурсів системи, чи ні. Висновок серверної частини передається клієнтській, яка й повідомляє про це користувача.<br>6. Необхідно передбачити неуспішне проходження аутентифікації при триразовому неправильному уведенні пароля.<br>Перевагою такої системи аутентифікації є те, що мережею передається лише хеш-образ пароля. Це підвищує стійкість системи аутентифікації по відношенню до перехоплення мережного трафіку.<br>Алгоритм хешування оберіть самостійно. |
| 3. Система шифрування текстових файлів. | 1. Створіть програму шифрування текстових файлів або файлів у форматі Microsoft Word за допомогою симетричного блочного криптоалгоритму.<br>2. Програма повинна приймати незашифрований файл на вході та повертати зашифрований файл. Аналогічно вона повинна приймати зашифрований файл та, розшифрувавши його, повертати відкритий текст.<br>3. Генерування ключів та обмін ключовою інформацією розробіть самостійно. |
| 4. Система шифрування | 1. Створіть простий мережний чат, що |

| | |
|---|---|
| мережного трафіка за допомогою потокового симетричного алгоритму. | використовує потокове шифрування мережного трафіка.<br>2. Для шифрування використайте один з потокових симетричних алгоритмів, доступних у CryptoAPI.<br>3. Довжину ключа та алгоритм обміну ключовою інформацією оберіть самостійно.<br>4. Програма повинна передавати уведену користувачем з клавіатури інформацію мережею у зашифрованому вигляді.<br>5. Програма повинна розшифровувати отриману мережею інформацію та виводити її на екран для читання користувачем. |
| 5. Система шифрування мережного трафіка за допомогою симетричного блокового алгоритму. | 1. Створіть простий мережний чат, що використовує блочне шифрування мережного трафіка.<br>2. Для шифрування використайте один з блочних симетричних алгоритмів, доступних у CryptoAPI.<br>3. Довжину ключа та алгоритм обміну ключовою інформацією оберіть самостійно.<br>4. Програма повинна передавати уведену користувачем з клавівтури інформацію мережею у зашифрованому вигляді.<br>5. Програма повинна розшифровувати отриману мережею інформацію та виводити її на екран для читання користувачем. |
| 6. Система шифрування текстових файлів за допомогою асиметричного криптоалгоритму. | 1. Створіть програму шифрування текстових файлів або файлів у форматі Microsoft Word за допомогою асиметричного криптоалгоритму.<br>2. Програма повинна приймати незашифрований файл на вході та повертати зашифрований файл. Аналогічно вона повинна приймати зашифрований файл та, розшифрувавши його, повертати відкритий текст..<br>3. Генерування ключів та обмін ключовою інформацією розробіть самостійно. |
| 7. Система електронного цифрового підпису для текстових файлів. | 1. Створіть програму, яка б виконувала процедуру цифрового підпису текстового файлу та перевірки цього підпису за допомогою CryptoAPI.<br>2. Програма повинна приймати текстовий файл з відкритою інформацією та виконувати процедуру електронного цифрового підпису |

| | |
|---|---|
| | цього файлу. Цифровий підпис повинен дописуватися у кінці файлу.<br>3. Програма повинна приймати текстовий файл з цифровим підписом та перевіряти його істинність.<br>4. Алгоритм електронного цифрового підпису та спосіб обміну ключовою інформацією оберіть самостійно. |
| 8. Система електронного цифрового підпису з хешуванням. | 1. Створіть програму, яка б виконувала процедуру електронного цифрового підпису хеш-образу відкритого тексту та перевіряла б істинність цього підпису.<br>2. Програма повинна приймати на вході файл з відкритим текстом, створювати хеш-образ та підписувати цей образ за допомогою одного з доступних алгоритмів ЕЦП.<br>3. Програма повинна приймати на вході файл з відкритим текстом, його хеш-образом та електронним цифровим підписом та перевіряти істинність цього підпису.<br>4. Алгоритми хешування та ЕЦП, довжину хеш-образу оберіть самостійно.<br>5. Алгоритм обміну ключовою інформацією також оберіть самостійно. |
| 9. Система аутентифікації на основі хешування парольної інформації (з використанням «токена безпеки») | 1. Пароль, що його уводять при реєстрації користувача в системі, хешується певним алгоритмом, до хеш-образу підмішується логін, отриманий образ запам‘ятовується у базі даних системи та прив‘язується до логіна.<br>2. Одночасно логін та пароль записуються на дискету або флеш-диск (обов‘язково на знімний носій!) у файли з жорстко визначеними іменами.<br>3. При спробі аутентифікації клієнтська частина повинна зчитувати логін та пароль тільки зі знімного носія.<br>4. Система повинна блокувати спроби користувача підмінити читання інформації зі знімного носія читанням з жорсткого диску; не можна також надати користувачеві можливості вибору файлів.<br>5. Клієнтська частина обчислює хеш-образ пароля та відправляє логін, і хеш-образ серверній частині.<br>4. Серверна частина вибирає з бази даних хеш- |

| | |
|---|---|
| | образ, що відповідає отриманому логіну, та порівнює його з отриманим образом.<br>5. В залежності від результату порівняння система робить висновок про те, допускати користувача до ресурсів системи, чи ні. Висновок серверної частини передається клієнтській, яка й повідомляє про це користувача.<br>Перевагою такої системи аутентифікації є те, що мережею передається лише хеш-образ пароля. Це підвищує стійкість системи аутентифікації по відношенню до перехоплення мережного трафіку.<br>Алгоритм хешування оберіть самостійно. |
| 10. Система шифрування текстових файлів за допомогою «токена безпеки» . | 1. Створіть програму шифрування текстових файлів або файлів у форматі Microsoft Word за допомогою симетричного блочного криптоалгоритму.<br>2. Програма повинна приймати незашифрований файл на вході та повертати зашифрований файл. Аналогічно вона повинна приймати зашифрований файл та, розшифрувавши його, повертати відкритий текст.<br>3. Ключова інформація повинна зберігатися на знімному носієві (дискета або флешка). При зашифруванні ключ повинен експортуватися на знімний носій (з неможливістю збереження його на жорсткому диску). При розшифруванні програма повинна зчитувати ключ тільки зі знімного носія. Необхідно заблокувати можливість вибору файлів з жорсткого диску. |

Якщо група налічує більше 10 студентів, кожен з них виконує завдання з цього ж набору, але з використання різних криптографічних алгоритмів. Конкретний алгоритм для кожного студента в такому разі визначає викладач.

### *Контрольні запитання*

1. Які функції виконує CryptoAPI?
2. Розкажіть про архітектуру CryptoAPI.
3. Охарактеризуйте криптографічні алгоритми, які Ви використали для Вашого завдання.
4. Які ще криптографічні алгоритми реалізує CryptoAPI?

### *Література*


1. Галицкий А.В., Рябко С.Д., Шаньгин В.Ф. Защита информации в сети. – М.:ДМК Пресс, 2004.
2. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа. – СПб.:Наука и техника, 2004.
3. Проскурин В.Г., Крутов С.В., Мацкевич И.В. Защита в операционных системах. – М.: «Радио и связь», 2000.
4. Щербаков А, Домашев А. Прикладная криптография. Использование и синтез криптографических интерфейсов. М.:Русская редакция, 2003.
5. М.А.Деднев, Д.В.Дыльнов, М.А.Иванов Защита информации в банковском деле и электронном бизнесе. М.:Кудиц-образ, 2004. – 512 с.