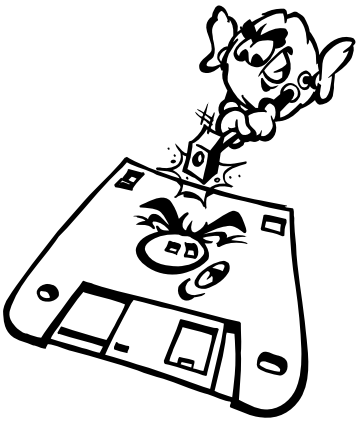
Кам'янець-Подільський національний університет

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт з дисципліни

«Захист комп'ютерної інформації»

студентами спеціальності 6.080200 «Прикладна математика»



Кам'янець-Подільський
2008

УДК 004.056 (075.8)
ББК 73.7 я73
М 54

Рецензенти:

О.М. Ткаченко, кандидат технічних наук, доцент кафедри обчислювальної техніки Вінницького національного технічного університету;

А.М. Кух, кандидат педагогічних наук, доцент кафедри МВФ і ДТОГ Кам'янець-Подільського національного університету.

М 54 **Методичні вказівки для виконання лабораторних робіт з дисципліни «Захист комп'ютерної інформації» студентами спеціальності 6.080200 «Прикладна математика»** / Автор-укл.: О.В. Слободянюк. – Кам'янець-Подільський: Кам'янець-Подільський національний університет, редакційно-видавничий відділ, 2008. – 62 с.

В посібнику наведено короткі теоретичні відомості, необхідні для розуміння основних моментів проблем захисту комп'ютерної інформації як на автономних комп'ютерних системах так і в комунікаційних мережах передавання даних. У рамках тем, що стосуються питань криптографії, розглянуто особливості деяких симетричних та асиметричних криптосистем із використанням програмного інтерфейсу Crypto API. Також у методичних вказівках подані основні відомості про поняття політики інформаційної безпеки та методику аудиту інформаційної системи установи на відповідність стандарту безпеки ISO 17799.

Рекомендується студентам спеціальностей «Прикладна математика», «Інформатика» та всім хто цікавиться питаннями захисту комп'ютерної інформації.

УДК 004.056 (075.8)
ББК 73.7 я73

Рекомендовано до друку вченою радою Кам'янець-Подільського національного університету, протокол № 10 від 29 листопада 2007 р.

ЗМІСТ

ВСТУП 4

Лабораторне заняття № 1. Інвентаризація ресурсів комп'ютерних робочих місць 6

Лабораторне заняття № 2. Захист від небажаних поштових повідомлень та програм-шпигунів 11

Лабораторне заняття № 3. Антивірусний захист комп'ютерних систем 14

Лабораторне заняття № 4. Криптографія. Симетричні системи шифрування 17

Лабораторне заняття № 5. Реалізація алгоритмів симетричного шифрування даних 24

Лабораторне заняття № 6. Crypto API. Асиметричні алгоритми шифрування даних 33

Лабораторне заняття № 7. Захист даних від модифікацій та підміни 41

Лабораторне заняття № 8. Захист комп'ютерних мереж 51

Лабораторне заняття № 9-10. Управління комп'ютерною безпекою 55

ЛІТЕРАТУРА 58

ДОДАТКИ 59

ВСТУП

Ще декілька років тому нові інформаційні та комп'ютерні технології були екзотикою у нашому житті. Наявність комп'ютера на підприємстві чи установі свідчило про його гарне фінансове становище, оскільки вартість звичайної робочої станції сягала астрономічних сум. Але вже на початку XXI століття людина у своїй діяльності все більше і більше стає залежною від «розумних залізячок». Виробництво, освіта, державні установи – скрізь комп'ютерна техніка стає надійним помічником, без якого вже дехто просто не уявляє свою роботу.

Сьогоднішній стан розвитку інформаційних технологій характеризується інтенсивною комунікацією пристроїв різних типів у інформаційні wan-, man- та lan-мережі. Запроваджуються нові стандарти фізичних середовищ передавання даних. Активно використовуються безпровідні технології комунікації пристроїв. Але в зв'язку з таким широким процесом інтеграції інформаційних та комунікаційних пристроїв виникає потреба обмеження доступу до них сторонніх осіб та незаконного втручання у їх роботу з метою перехоплення та викрадення цифрових даних, що являють собою певну цінність. Тому актуальність питання захисту інформації у даній сфері надзвичайно висока.

За даними статистики, ще п'ять років тому гостро проявлялась нестача кваліфікованих адміністраторів. Зараз ситуація докорінно змінилася. Забезпечення кадрами, які здатні підтримувати у робочому вигляді програмний та технічний стан комп'ютерної техніки є більш менш задовільним. Але тепер на перший план виходить проблема саме захисту створених державними установами чи приватними компаніями інформаційних систем та мереж. А для цього необхідні спеціально підготовлені спеціалісти – офіцери інформаційної безпеки. Звичайно підготовка такого висококваліфікованого фахівця у сфері захисту цифрових даних надзвичайно складна справа, та все ж володіти основними поняттями і найпростішими методиками боротьби із незаконними втручаннями у роботу автоматизованих комп'ютерних систем повинен кожен сучасний спеціаліст у сфері IT-технологій.

Дані методичні вказівки призначені допомогти студентам оволодіти основними прийомами та навичками по боротьбі із фактами порушення комп'ютерної безпеки автоматизованих робочих місць та вміти приймати першочергові запобіжні заходи. Коло запропонованих питань включає собою як практичні способи самостійного візуального обстеження комп'ютерної системи так і використання спеціалізованих програмних засобів по виявленню та боротьбі із шкідливими програмами-закладками та комп'ютерними вірусами. Крім цього, під час виконання лабораторних робіт студенти будуть мати змогу ознайомитися із криптографічними методами захисту цифрових даних. А також отримають основні поняття про політику інформаційної безпеки організації, особ-

ливості її розробки та ознайомляться із методикою аудиту інформаційної системи установи на відповідність стандарту ISO 17799 за допомогою спеціальних програмних пакетів.

Для виконання лабораторних робіт викладачу бажано підготувати спеціальне автономне робоче місце, яке не буде зв'язане жодним комунікаційним каналом з іншими або створити віртуальні копії операційних систем за допомогою програм на зразок VMWare або Microsoft VirtualPC.

Лабораторне заняття № 1

ІНВЕНТАРИЗАЦІЯ РЕСУРСІВ КОМП'ЮТЕРНИХ РОБОЧИХ МІСЦЬ

Мета: Навчитися проводити повний аналіз комп'ютерних місць на предмет виявлення можливих «дірок» у безпеці апаратних та програмних засобів.

Необхідне програмне забезпечення

OS Windows, Everest, mvPCinfo, Hinfo, довідник процесів, довідник по реєстру ОС Windows.

Теоретична частина

Успіх реалізації того або іншого алгоритму атаки хакера на практиці в значній мірі залежить від архітектури і конфігурації конкретної операційної системи, що є об'єктом цієї атаки. Проте є атаки, яким може бути піддана практично будь-яка операційна система:

- *крадіжка пароля;*
- *підглядання за користувачем,* коли той вводить пароль, що дає право на роботу з операційною системою;
- *отримання пароля з файлу,* в якому цей пароль був збережений користувачем;
- *пошук пароля,* який користувачі, щоб не забути, записують на календарях, в записниках і т.д.;
- *крадіжка зовнішнього носія парольної інформації;*
- *повний перебір всіх можливих варіантів пароля* (brute force – груба сила);
- *підбір пароля по частоті повторень символів;*
- *сканування жорстких дисків комп'ютера;*
- *збір «сміття»;*
- *перевищення повноважень;*
- *запуск програми від імені користувача*, що має необхідні повноваження, або як системна програма;
- *підміна динамічної бібліотеки*, що використовується системними програмами;
- *модифікація коду або даних підсистеми захисту самої операційної системи;*
- *відмова в обслуговуванні* (метою цієї атаки є часткове або повне виведення з ладу операційної системи);
- *захоплення ресурсів* (програма хакера здійснює захоплення всіх ресурсів, що є в операційній системі, а потім входить в нескінченний цикл);
- *бомбардування запитами* (програма хакера постійно направляє операційній системі запити, реакція на які вимагає залучення значних ресурсів комп'ютера);

- *використання помилок в програмному забезпеченні або адмініструванні.*

Якщо в програмному забезпеченні комп'ютерної системи немає помилок і її адміністратор суворо дотримується політики безпеки, яка рекомендується розробниками операційної системи, то атаки всіх перерахованих типів малоефективні.

Перераховані вище методи атаки хакера на комп'ютерну систему є найтиповішими і описані в загальній формі. Для узагальненої моделі НСД та взлому комп'ютерних систем можна сформулювати універсальні правила, згідно яких, щоб звести ризик до мінімуму, потрібно:

- *Не відставати* від хакерів: потрібно бути завжди в курсі останніх розробок з області комп'ютерної безпеки. Необхідно оформити підписку на декілька спеціалізованих журналів, в яких детально освітлюються питання захисту комп'ютерних систем від взлому. Регулярно проглядайте матеріали, що поміщаються на серверах хакерів Internet (наприклад, astalavista.box.sk, xakep.ru).
- *Керуватися* принципом розумної достатності: не потрібно намагатися побудувати абсолютно надійний захист. Адже чим надійніший захист, тим більше він споживає ресурсів комп'ютерної системи і тим важче користуватися ним.
- *Зберігати* в таємниці інформації про принципи дії захисних механізмів комп'ютерної системи. Чим менше хакеру відомо про ці принципи, тим важче буде для нього організувати успішну атаку.
- *Намагатися* максимально обмежити розміри комп'ютерної мережі, що знаходиться під захистом, і без крайньої необхідності не допускати її підключення до Internet.
- *Пошукати* інформацію про нове програмне забезпечення, що є на серверах хакерів Internet, перш ніж вкласти кошти в його закупку.
- *Розміщувати* сервери в приміщеннях, що знаходяться під охороною. Не потрібно підключати до них клавіатуру і монітори, щоб доступ до цих серверів здійснювався тільки через мережу.
- *Шифрувати* і забезпечувати цифровим підписом абсолютно всі повідомлення, що передаються по незахищених каналах зв'язку, повинні.
- *Пропускати* всі повідомлення через брандмауер, *шифрувати* і *оснащати їх цифровим підписом*, якщо вони відправляються або отримуються з незахищеного каналу в захищений і навпаки.
- *Не нехтувати* можливостями, які надає аудит. Інтервал між сеансами перегляду журналу аудиту не повинен перевищувати однієї доби.
- *Вивчати* уважно всі нові записи з журналу системного аудиту особливо коли їх кількість різко зростає, що може свідчити про наявність атаки хакера, який намагається приховати сліди свого втручання в роботу даної комп'ютерної системи.
- *Регулярно проводити* перевірку цілісності програмного забезпечення комп'ютерної системи. Перевіряти її на наявність програмних закладок.

- *Реєструвати* всі зміни політики безпеки в звичайному паперовому журналі. Регулярно звіряйте політику безпеки із зареєстрованою в цьому журналі. Це допоможе вказати на присутність програмної закладки, якщо вона була впроваджена хакером в комп'ютерну систему.
- *Користуватися* операційними системами із вбудованою системою безпеки.
- *Створити* декілька пасток для хакерів (наприклад, створіть на диску файл з привабливим ім'ям, прочитати який неможливо за допомогою стандартних засобів, і якщо буде зафіксоване успішне звернення до цього файлу, значить, в комп'ютерну систему, що знаходяться під захистом, була впроваджена програмна закладка).
- *Регулярно тестувати* комп'ютерну систему за допомогою спеціальних програм, призначених для визначення ступеня її захищеності від атак хакерів.

Практичне завдання

1. Проведіть інвентарний облік свого робочого комп'ютерного місця.
    1) Апаратна частина.
        а) Наявність зовнішніх ушкоджень основних комплектуючих (системний блок, монітор, клавіатура…).
        б) Монітор (розмір по діагоналі, виробник, модель).
        в) Системний блок:
            - Процесор (виробник, модель, тактова частота)
            - Оперативна пам'ять
            - Материнська плата
            - Відеокарта
            - Мережева карта
            - Магнітооптичні пристрої (CD-ROM/RW, DVD-ROM/RW, FDD, ZIP…)
            - Жорсткі диски (виробник, модель, швидкість обертання пластин, розмір кеш-пам'яті, інтерфейс передачі даних)
            - Комунікаційні порти (кількість кожного наявного типу)
            - Інші (звукова карта, модем, ТВ-тюнер, RAID-контролер…)
        г) Периферійні пристрої:
            - Принтер
            - Сканер
            - ДБЖ (UPS)
            - Клавіатура
            - «Миша»
            - Інші

    2) Програмне забезпечення.

    Перевірте встановлене програмне забезпечення (назва, версія, серійний номер)

- BIOS
- Операційна система (виробник, версія, IP-адреса, домен, пароль адміністратора…)
- Необхідні драйвера пристроїв (відсутні, неробочі…)
- «Офісний» набір: текстовий редактор, редактор ЕТ, редактор презентацій, СУБД, графічний переглядач (редактор)
- Електронна пошта
- Браузер

3) Документальний облік.

- Наявні договори закупки програмного забезпечення
- Акти передачі матеріально-технічних засобів
- Посадові інструкції підзвітного відділу
- Правила та інструкції з питань безпеки та охорони праці

Для збору даних про технічні характеристики комплектуючих системного блоку можна скористатися спеціалізованими утилітами ***MvPCinfo***, ***Everest***, ***Hinfo***, ***Aida32*** або стандартними засобами операційної системи.

В разі виявлення підозрілого або невідомого обладнання зробити відповідну помітку.

2. Перевірте операційну систему на наявність підозрілого програмного забезпечення.

1) За допомогою оснастки ***msconfig*** та редактору реєстру ***regedit*** переглянути список автозавантаження.

Звернути увагу на наступні ключі реєстру:

1. [HKLM\Software\Microsoft\Windows\ CurrentVersion\Run]
2. [HKLM\Software\Microsoft\Windows\ CurrentVersion\RunOnce]
3. [HKLM\Software\Microsoft\Windows\ CurrentVersion\RunServices]
4. [HKLM\Software\Microsoft\Windows\ CurrentVersion\RunServicesOnce]
5. [HKCU\Software\Microsoft\Windows\ CurrentVersion\Run]
6. [HKCU\Software\Microsoft\Windows\ CurrentVersion\RunOnce]
7. [HKCR\exefile\shell\open\command]
8. [HKLM\Software\Microsoft\ActiveSetup\ InstalledComponents\ KeyName]
9. [HKLM\SOFTWARE\Classes\exefile\shell\open\command]

2) Переглянути список запущених процесів Windows та визначити серед них підозрілі (незнайомі). Для визначення таких процесів можна скористатися довідником стандартних процесів Windows «services.chm», який можна вільно знайти у мережі інтернет.

3) Провести «поверхневий» огляд файлової структури всіх дисків.
4) Знайдені підозрілі програмні продукти занести в звіт.

3. Проведіть зачистку «сміття» у всіх можливих місцях його накопичення.
Провести аналіз наявних файлів «сміття» у наступних папках:

1) Папки тимчасового зберігання файлів всіх користувачів (…Document and Settings\UserName\Local Settings\Temp).
2) Папки об'єкту Windows «Корзина» на всіх дисках (…\RECYCLER).
3) Папка тимчасового зберігання системних файлів операційної системи (…\Windows\Temp).

Контрольні запитання

1. На які групи можна поділити спроби несанкціонованого доступу до комп'ютерних систем?
2. Що розуміють під поняттями загроза, атака та порушення?
3. Хто такі зовнішні та внутрішні порушники?
4. Яку загрозу несуть собою порушники?
5. Сліди діяльності якого класу порушників найважче виявити?
6. Які заходи потрібно проводити задля запобіганню несанкціонованого доступу?
7. Для чого потрібно проводити регулярний огляд комп'ютерних систем?
8. Що дає повна інвентаризація програмних засобів?
9. Що саме фіксується в журналі аудиту?
10. Якими методами користуються зловмисники при підборі ключів?

Лабораторне заняття № 2

ЗАХИСТ ВІД НЕБАЖАНИХ ПОШТОВИХ ПОВІДОМЛЕНЬ ТА ПРОГРАМ-ШПИГУНІВ

Мета: ознайомлення з методами боротьби із програмами-шпигунами та захисту від небажаної електронної кореспонденції.

Необхідне програмне забезпечення

ArgoSoft MailServer, The Cleaner, Ad-Ware, Ms Antispyware, Advanced Mail Sender

Теоретична частина

ПРОГРАМНІ ЗАКЛАДКИ

Головною умовою правильного функціонування такої комп'ютерної системи є забезпечення захисту від втручань в процес обробки інформації тих програм, присутність яких в комп'ютерній системі не є обов'язковою. Серед подібних програм, в першу чергу, слід виділити комп'ютерні віруси. Однак існують шкідливі програми ще одного класу. Від них, як і від вірусів, слід з особливою ретельністю очищати свої комп'ютерні системи. Це так звані програмні закладки, які можуть виконувати хоча б одну з перерахованих нижче дій:

- вносити довільні спотворення в коди програм, що знаходяться в оперативній пам'яті комп'ютера (програмна закладка першого типу);
- переносити фрагменти інформації з одних областей оперативної або зовнішньої пам'яті комп'ютера в інші (програмна закладка другого типу);
- спотворювати інформацію, що виводиться на зовнішні комп'ютерні пристрої або в канал зв'язку, отриману в результаті роботи інших програм (програмна закладка третього типу).

Завдання захисту від програмних закладок може розглядатися в трьох принципово різних варіантах:

- не допустити можливості занесення програмної закладки в комп'ютерну систему;
- виявити занесену програмну закладку;
- видалити занесену програмну закладку.

ЗАХИСТ ВІД НЕБАЖАНОЇ ЕЛЕКТРОННОЇ ПОШТИ

В Internet словом **спам** називають небажанні електронні повідомлення відправлені за індивідуальною електронною адресою або в групу новин. Така пошта відома ще як «макулатурна пошта», або UCE-повідомлення (unsolicited commercial e-mail – небажані повідомлення комерційного характеру). Зазвичай такі повідомлення містять сумнівні рекламні оголошення, методи швидкого збагачення або навіть непристойні пропозиції сексуального характеру, сло-

вом те, що ви зовсім не бажаєте читати. Такі повідомлення називаються спамом, практика розсилки цих повідомлень – **спамінгом**, а людина, яка їх розсилає, – **спамером**.

Задача захисту від спаму полягає у створенні набору правил доставки для блокування небажаних повідомлень. Допомогу у цьому процесі покликані надавити спеціальні програми-фільтри спаму. До найбільш відомих можна віднести McAfee SpamKiller, Antispam, набори фільтрів для поштових клієнтів BayesIt, Spamihilator та інші. Їх можна розподілити на дві категорії. Перші перевіряють заголовки та адреси відправників поштових повідомлень перед завантаженням її до поштового клієнта користувача, а другі проводять аналіз вмісту вже завантажених повідомлень.

Практичне завдання

Моделювання ситуації № 1

Вас прийняли на посаду системного адміністратора із виконанням обов'язків офіцера комп'ютерної безпеки. Ваш попередник був звільнений за «нечистоплотність» в роботі. Тому першим вашим завданням стає пошук можливих «сюрпризів» від нього. Насамперед цими сюрпризами можуть виявитися програмні закладки, які дозволяють створювати дірки у системі захисту комп'ютерної системи, та троянські програми, які зможуть продовжувати збирати конфіденційну інформацію про вашу фірму.

Завдання:

1) Проведіть поверхневий огляд можливих місць присутності програм-закладок та троянських програм. Спробуйте власноруч виявити ознаки наявності програм-шпигунів в комп'ютерній системі.
2) Встановіть одну із спеціалізованих програм для боротьби із шпигунськими програмами та проведіть за їх допомогою сканування системи.
3) Детально занотуйте результати свого «ручного» огляду та «автоматичного» сканування програм.
4) Подайте в звіті кількість знайдених програм-шпигунів, їх назви та виявлені місця «дислокації» на жорсткому диску чи в об'єктах операційної системи.

Моделювання ситуації № 2

Адреса вашої поштової корпоративної скриньки потрапила до списків спам-розсилок. Внаслідок цього кожного дня ваша компанія несе великі збитки внаслідок того, що для перегляду та сортування поштових повідомлень витрачається значна кількість часу та коштів. Знайдіть вирішення даного питання шляхом використання спам-фільтрів.

Завдання:

1) Змоделюйте за допомогою програмного комплексу Advanced Mail Sender та Argosoft Mail Server процес розсилки та отримання небажаних рекламних листів для декількох користувачів (більше 10).
2) Із запропонованого викладачем анотованого списку виберіть та застосуйте антиспамову програму.
3) Спробуйте домогтися абсолютного результату (100% блокування небажаних повідомлень).
4) Всі кроки та результати своїх пошуків занесіть у звіти.

Контрольні запитання

1. Що таке програмна закладка?
2. Які ознаки проникнення програмної закладки в систему?
3. Які умови повинні виконуватися для активізації програмних закладок?
4. Яку загрозу являють собою троянські програми?
5. Що таке спам?
6. Які методи боротьби зі спамом вам відомі?
7. Які програмні засоби захисту від спаму ви знаєте?
8. Що таке спам-фільтри?
9. Який принцип роботи спам-фільтру?
10. Яка різниця між комп'ютерним вірусом і програмною закладкою?

Лабораторне заняття № 3

АНТИВІРУСНИЙ ЗАХИСТ КОМП'ЮТЕРНИХ СИСТЕМ

Мета: Оволодіти навичками та знаннями по боротьбі із комп'ютерними вірусами та профілактиці зараження ними комп'ютерних систем.

Необхідне програмне забезпечення

Набір «дієздатних» вірусів, антивірусні програмний комплекс Symantec Antivirus Corporate Edition.

Теоретична частина

ПОНЯТТЯ ВІРУСУ

Історично **вірусом** називається будь-яка програма, що заражає виконувані або об'єктні файли. Програму, відтворюючу себе без відома користувача, також можна віднести до вірусів. Найчастіше вірус поміщає своє тіло в програмному файлі так, щоб він активізувався при кожному запуску програми. Крім того, віруси можуть вражати завантажувальний сектор жорсткого або гнучкого диску, поміщеного в дисковод зараженого комп'ютера. Перенесення свого тіла на дискети і жорсткі диски є для вірусу гарантією того, що він буде запущений при кожному наступному включенні системи.

Більшість вірусів інфікують файли шляхом перенесення свого тіла всередину жертви. Проте у зв'язку із створенням все більш досконалих антивірусних програм, розробники вірусів змінили стратегію. Тепер, перш ніж заразити черговий файл, вірус змінює своє тіло. Вибравши жертву, вірус копіює себе з пам'яті комп'ютера всередину файлу, що заражається.

Найбільш поширеними типами вірусів є троянські коні, поліморфні і неполіморфні віруси, що шифруються, стелс-віруси, повільні віруси, ретро-вирусі, складені віруси, озброєні віруси, віруси-фаги і макровіруси.

Деякі віруси не можуть бути виявлені навіть найкращими антивірусними програмами. Як додаткові засоби захисту потрібно залучити освіченість користувача. Бажано вміти виявляти орієнтовні ознаки зараження:

- Програми стали завантажуватися повільніше.
- Файли з'являються або зникають.
- Розміри програми або об'єкту змінюються з незрозумілих причин.
- На екрані з'являється незвичайний текст або зображення.
- Елементи екрану виглядають нечіткими, розмитими.
- Сам по собі зменшується об'єм вільного місця на жорсткому диску.
- Команди CHKPSK і SCANDISK повертають некоректні значення.
- Імена файлів змінюються без видимих причин.
- Натиснення клавіш супроводжуються дивними звуками.

- Доступ до жорсткого диска заборонений.

Для того щоб захистити свою комп'ютерну систему від вірусів, виконуйте із всіма вхідними даними наступні три операції.

По-перше, перевіряйте всю інформацію, що приходить, за допомогою антивірусних програм. І неважливо, як ця інформація потрапила до вас: на дискетах або в повідомленнях електронної пошти. Тільки після цього можна запускати одержані програми або проглядати документи, що прийшли.

По-друге, поставте антивірусне програмне забезпечення прямо на комп'ютері-брандмауері так, щоб заражені файли не могли проникнути в мережу. Більшість з них містить готові антивірусні програми, а також засоби перевірки цілісності даних. З їх допомогою можна проглядати у реальному часі всі зміни в системі. Так ви зможете помітити більшість операцій, що виконуються вірусами. Крім того, більшість брандмауерів надають засоби захисту DOS-сеансів.

По-третє, встановіть антивірусне програмне забезпечення на кожній підключеній до мережі робочих станцій. Так ви зможете запобігти розповсюдженню вірусів по мережі, якщо одна з машин буде заражена.

Практичне завдання

1. Забезпечити антивірусним захистом віртуальну локальну мережу за допомогою Symantec Antivirus Corporate Edition.

Розгортання комплексу антивірусного захисту Symantec Antivirus розпочніть із встановлення його серверної частини під операційною системою Windows 2000 Server. При встановленні серверної частини комплексу необхідно дотримуватися такої послідовності дій:

1) Встановіть консоль управління Symantec System Center.
2) За допомогою даної консолі встановіть безпосередньо сам сервер антивірусу.
3) У створеній групі антивірусній серверів зробіть ваш сервер первинним (меню «Сервис» в консолі SSC).

Якщо все було зроблено правильно, у вікні консолі управління в пункті «Структура системы» має з'явитися створена вами група серверів. Виділіть її і виконайте команду меню «Сервис» а «Удаленная установка клиента». Виконуйте всі вимоги майстра і встановіть на необхідну робочу станцію клієнтську частину.

2. Самостійно налаштуйте автоматичне обновлення «клієнта» із локального серверу.

3. Проведіть повну перевірку віддаленого «клієнта» та занотуйте всі знайдені антивірусом підозрілі об'єкти у наступну таблицю:

№ з/п	Інфікований об'єкт	Назва вірусу, його можливий тип (клас), за якою класифікацією (Symantec, Panda Software, Kaspersky Lab…)	Результат дії антивірусу
1			

Контрольні запитання

1. Що таке сигнатура вірусу?
2. Які існують типи вірусів?
3. Які ознаки зараження вірусом комп'ютерної системи?
4. Що таке віруси-фаги?
5. Що таке макровіруси?
6. Які основні шляхи розповсюдження вірусів?
7. Які основні складові модулі програм-антивірусів?
8. Чому поліморфні віруси найнебезпечніші?
9. Що таке комп'ютерна вакцина?
10. Назвіть сучасні антивірусні програмні засоби.

Лабораторне заняття № 4

КРИПТОГРАФІЯ. СИМЕТРИЧНІ СИСТЕМИ ШИФРУВАННЯ

Мета: Познайомитися із методами кодування і криптографічного захисту інформації та реалізувати алгоритми роботи деяких методів.

Необхідне програмне забезпечення

Компілятор мови програмування (на вибір викладача C++, Basic, Pascal...)

Теоретична частина

ОСНОВНІ ПОНЯТТЯ І ТЕРМІНИ

Криптографія (cryptography) (гр. ktyptos – таємний і grapho – пишу) – наука про способи перетворення (шифрування) інформації з метою захисту її від незаконного чи небажаного використання. Шифрування повинно також забезпечувати можливість легкого одержання вихідної інформації із зашифрованої легітимними користувачами, яким відомий чи доступний ключ алгоритму розшифровування або інша ключова інформація. Найчастіше передбачається, що зашифрована інформація передається по загальнодоступному каналу зв'язку, наприклад, радіо чи Інтернет, так, що всі користувачі (законні і незаконні) мають до неї вільний доступ. У математичній чи теоретичній криптографії прагнуть показати, що задача зламування шифру є задачею із строго доказовими властивостями, зокрема, її розв'язування має визначену «складність» у рамках тієї чи іншої математичної моделі.

Шифрування (encryption) – процес зашифровування інформації, тобто застосування криптографічного перетворення до вхідних даних (відкритого тексту).

Дешифрування (deciphering) і ***розшифровування (decryption)*** – відповідно методи одержання із зашифрованого тексту інформації без знання криптографічного ключа і зі знанням його.

Шифр і ***шифросистема (cipher, cypher, ciphercode)*** – найчастіше вихід криптосистеми і сама симетрична криптосистема відповідно. У залежності від контексту шифр може означати зашифроване повідомлення або саму криптографічну систему перетворення інформації.

Криптографічний ключ (cryptographic key, іноді просто ***key***) – для симетричних криптосистем – секретний компонент шифру. Повинен бути відомим тільки законним користувачам процесу обміну інформації.

Цикл шифрування (round) – один комплексний крок алгоритму, в процесі якого перетворюються дані.

Гама-послідовність чи просто ***гама (gamma sequence, gamma)*** – послідовність псевдовипадкових елементів, що генеруються за певним законом чи алгоритмом.

Гамування (gamma xoring) – процес «накладання» гама-послідовності на відкриті дані, найчастіше з допомогою логічної операції XOR (виключне АБО).

Імітозахист – це захист даних у системах передачі і зберігання від спотворення чи зловмисного нав'язування помилкової інформації. Імітозахист досягається найчастіше за рахунок включення в пакет переданих даних імітовставки.

Імітовставка – блок додаткової інформації, обчислений за певним алгоритмом і залежний від деякого криптографічного ключа і даних.

Криптоаналіз (cryptanalysis) – набір методик і алгоритмів дешифрування криптографічно захищених повідомлень, аналізу шифросистем.

Симетрична криптосистема – система із секретним ключем, який використовується і під час шифрування, і під час розшифровування. Для одержання секретного ключа учасниками обміну інформацією потрібен секретний канал, що є головним недоліком класичних симетричних криптосистем.

Асиметрична криптосистема чи система з відкритим (публічним) ключем – система із двома ключами. Генеруються два ключі – секретний і відкритий. Відкритий ключ передається відкритими каналами зв'язку одному або декільком партнерам, з допомогою нього шифруються повідомлення і передаються володарю секретного ключа. Розшифрувати повідомлення можна тільки секретним ключем. Знайти секретний ключ за відомим публічним – обчислювально складна задача, яку неможливо розв'язати (поки що) за допомогою комп'ютерних систем. Для генерації ключів і шифрування використовуються обчислювально-складні алгоритми, наприклад, розклад астрономічно великих чисел на прості множники, цілочисельне логарифмування, еліптичні функції. Алгоритми, які ґрунтувались на задачі вкладання ранця, виявились ненадійними і спростовані математично. Недоліком асиметричних криптосистем є їх нестроге теоретичне об ґрунтування. Однак є практичні області, де без них обійтись неможливо – це системи електронного підпису, електронного голосування, «електронні» гроші.

Для електронного підпису автор шифрує документ секретним ключем. Перевірка підпису відбувається з допомогою відкритого ключа, тобто відбувається процес, протилежний до описаного вище. Для запобігання внесення змін, сторони, що підписують договір, зашифровують документ послідовно кожен своїм секретним ключем.

ПЕРЕСТАНОВОЧНІ ШИФРИ

Простий стовпчиковий перестановочний шифр

У даному виді шифру текст пишеться на горизонтально розграфленому аркуші паперу фіксованої ширини, а шифротекст прочитується по вертикалі. Дешифрування полягає в записі шифротексту вертикально на аркуші розграф-

леного паперу фіксованої ширини і потім зчитуванні відкритого тексту горизонтально.

Відкритий текст: КАМЯНЕЦЬ-ПОДІЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Зашифрований текст: КЦИЖ_РК АЬЛЙАУСЦ М-Ь_ВНИІ ЯПСДНІТИ НОЬЕИВЕЖ ЕДКРЙЕТ

К	А	М	Я	Н	Е
Ц	Ь	-	П	О	Д
І	Л	Ь	С	Ь	К
И	Й	_	Д	Е	Р
Ж	А	В	Н	И	Й
_	У	Н	І	В	Е
Р	С	И	Т	Е	Т

Перестановочний шифр із ключовим словом

Літери відкритого тексту записуються в комірки прямокутної таблиці по її рядках. Літери ключового слова пишуться над стовпцями і вказують порядок цих стовпців (за збільшенням номерів літер в алфавіті). Щоб отримати зашифрований текст, треба виписувати літери по стовпцях з урахуванням їх нумерації:

Ш	и	ф	р
4	1	3	2
П	р	и	к
л	а	д	н
а		м	а
т	е	м	а
т	и	к	а

Відкритий текст: *Прикладна математика* Ключ: *Шифр*

Криптограма: Ра икнаааидммкплатт

Ключове слово (послідовність стовпців) відоме адресату, який легко зможе розшифрувати повідомлення.

Оскільки символи криптотекста ті ж, що і у відкритому тексті, то частотний аналіз покаже, що кожна літера зустрічається приблизно з тією ж частотою, що і звичайно. Це дає криптоаналітику інформацію про те, що це перестановочний шифр. Застосування до криптотексту другого перестановочного фільтру значно підвищить безпеку. Існують і ще складніші перестановочні шифри, але із застосуванням комп'ютера можна розкрити майже всі із них.

Хоча багато сучасних алгоритмів використовують перестановку, з цим пов'язана проблема використання великого об'єму пам'яті, а також іноді потрібна робота із повідомленнями певного розміру. Тому частіше використовують підстановочні шифри.

ПІДСТАНОВОЧНІ ШИФРИ

У підстановочних шифрах літери вхідного сполучення замінюються на підстановки. Заміни в криптотексті розташовані в тому ж порядку, що і в оригіналі. Якщо використання замін не змінюється протягом всього тексту, то криптосистема називається одноалфавітною (моноалфавітною). У багатоалфавітних системах використання підстановок змінюється в різних частинах тексту.

Шифр Цезаря

Юлій Цезар оповідає про відсилання зашифрованого повідомлення Цицерону. Використана при цьому система підстановок була одноалфавітною,

але не являлася системою Цезаря: латинські літери замінювалися на грецькі способом, який не був ясний із розповіді Цезаря. Інформація про те, що Цезар дійсно використовував систему Цезаря, прийшла від Светонія.

У шифрі Цезаря кожна літера заміщається на літеру, що знаходиться k символами правіше по модулю рівній кількості літер в алфавіті. (Згідно Светонію в Цезаря k = 3 n = 50)

$C_k(j)=(j+k)(\text{mod } n)$, n – кількість літер в алфавіті

Очевидно, що зворотною підстановкою є

$$C_k^{-1}(j) = C_{n-k} = (j + n - k) \text{ (mod } n)$$

Афінна криптосистема

Узагальненням системи Цезаря є афінна криптосистема. Вона визначається двом числами a і b, де 0 <= a, b <= n – 1. n – як і раніше, є міцністю алфавіту. *Числа a і n* повинні *бути взаємно прості.*

Відповідними замінами є:

$$A_{a,b}(j) = (a*j + b) \text{ (mod } n)$$
$$A^{-1}_{a,b}(j) = (j - b)*a^{-1} \text{ (mod } n)$$

Зворотну заміну також можна отримати, просто помінявши місцями рядки в таблиці замін. Взаємна простота a і n необхідна для бієктивності відображення, інакше можливі відображення різних символів в один і неоднозначність дешифрування.

ШИФР ПОЛІБІЯ

	А	**Б**	**В**	**Г**	**Д**	**Е**
А	А	Б	В	Г	Д	Е
Б	Є	Ж	З	И	І	Ї
В	Й	К	Л	М	Н	О
Г	П	Р	С	Т	У	Ф
Д	Х	Ц	Ч	Ш	Щ	Ь
Е	Ю	Я	.	,	-	

Система Цезаря не є найстарішою. Можливо, що найбільш древньою з відомих є система грецького історика Полібія, який помер за 30 років до народження Цезаря. Його суть полягає в наступному: розглянемо прямокутник, який ще часто називають дошкою Полібія.

Кожна літера може бути представлена парою літер, які вказують на рядок і стовпець, в яких розташована дана літера. Так представлення літер В, Г, П, У будуть АВ, АГ, ВГ ГБ відповідно, а повідомлення ПРИКЛАДНА МАТЕМАТИКА зашифрується як

ВГВДБВБДБЕАААДВБААЕБЕЕВАААГААЕВАААГАБВБДААЕЕ

БАГАТОАЛФАВІТНІ СИСТЕМИ

Поліалфавітні підстановочні шифри були винайдені Ліном Баттістою (Lean Battista) в 1568 році. Основна ідея багатоалфавітних систем полягає в тому, що впродовж всього тексту одна і та ж літера може бути зашифрована по-різному. Тобто заміни для літери вибираються із багатьох алфавітів залежно від положення в тексті. Це є хорошим захистом від простого підрахунку частот, тому що не існує єдиного маскування для кожної літери в криптотексті. У даних

шифрах використовуються множинні однобуквені ключі, кожний з яких використовується для шифрування одного символу відкритого тексту. Першим ключем шифрується перший символ відкритого тексту, другим – другий, і т.д. Після використання всіх ключів вони повторюються циклічно.

Xor-алгоритм

Найпростішим в реалізації різновидом поліалфавітного шифру є XOR-алгоритм.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Операція XOR – eXclusive OR, виключне АБО чи додавання за модулем 2 – результат логічної операції з двома двійковими розрядами дорівнює 1, якщо розряди різні, і дорівнює 0, якщо розряди співпадають.

Повторне застосування операції XOR відновлює початкове значення, внаслідок чого алгоритм можна використовувати і для шифрування і для розшифровування.

Шифр Віженера

Однією із старих і найбільш відомих багатоалфавітних криптосистем є система Віженера, названа на честь французького криптографа Блейза Віженера (Vigenere), в математиці даний шифр називається шифром Трітеміуса. Цей метод був вперше опублікований в 1586 році. У даному шифрі ключ задається набором з d літер. Такі набори підписуються із повторенням під повідомленням, а потім отриману послідовність складають із відкритим текстом по модулю n(потужність алфавіту). Тобто виходить наступна формула:

$$Vig_d(m_i) = (m_i + k_{i \bmod d}) \pmod n$$

Також літеру шифротексту можна знаходити із наступної таблиці, як перетин стовпця, визначуваного літерою відкритого тексту, і рядка, визначуваною літерою ключа (див. додаток 2).

В окремому випадку, при d = 1, отримуємо шифр Цезаря.

Шифр Бофорта

Іншим різновидом шифру Віженера є шифр Бофорта, названий на честь адмірала сера Френсіса Бофорта – творця шкали для визначення швидкості вітру. Його рядками є рядки квадрата Віженера, записані в зворотному порядку, а перша і остання рядки поміняні місцями (див. додаток 3).

Формулою перетворення буде:

$$Bof_d(m_i) = (k_{i \bmod d} - m_i) \pmod n$$

В обох випадках зворотна підстановка легко визначається з квадрату, або за формулами

$$Vig_d^{-1}(m_i) = (m_i - k_{i \bmod d}) \pmod n$$
$$Bof_d^{-1}(m_i) = Bof_d(m_i) = (k_i - m_{i \bmod d}) \pmod n$$

Повторне застосування двох або більш шифрів Віженера називатиметься *складеним шифром Віженера*. Він має рівняння

$$Vig^*(m_i) = (m_i + k_{i \bmod dk} + l_{i \bmod dl} + ... + s_{i \bmod ds}) \pmod n$$

де $k_i + l_i + ... + s_i$ взагалі кажучи, мають різні періоди dk, dl, ..., ds відповідно. Період їх суми $k_i + l_i + ... + s_i$ буде найменшим спільним кратним окремих періодів.

Якщо ключ k не повторюється, то вийде *шифр Вернама*. Якщо в якості ключа використовується текст, що має смисл, то маємо шифр «біжучого ключа».

Шифр Віженера із перемішаним один раз алфавітом

Такий шифр представляє собою просту підстановку із подальшим застосуванням шифру Віженера:

$$Vig\hat{}(m_i) = f(m_i) + k_{i \bmod d},\ \ Vig\hat{}^{-1}(m_i) = f^{-1}(m_i - k_{i \bmod d}).$$

Для шифрів Віженера та Бофорта існують таблиці, за якими можна проводити шифрування та дешифрування повідомлень (додаток 2 та 3).

Практичне завдання

1. Зашифрувати фразу за допомогою перестановочного шифру із ключовим словом (кількість літер від 4 до 8).
2. Реалізувати алгоритм (блок-схема, семантична діаграма, мова програмування) шифрування даних за допомогою дошки Полібія.
3. Зашифрувати фразу із використанням модернізованого коду Цезаря (змінне число k).
4. Використовуючи таблицю Віженера для українського алфавіту зашифрувати фразу згідно свого варіанту.
5. Використовуючи таблицю Бофорта для українського алфавіту дешифрувати фразу згідно свого варіанту.

Примітки

- У другому завданні використати такий алфавіт (кількість n = 34): ***А Б В Г Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я _ «Символ пробілу»***
- Варіанти даних для шифрування за варіантами можна отримати із наступної таблиці.

Варіант	Фраза для шифрування	Зміщення k (для коду Цезаря)	Фраза для дешифрування	Ключ для шифрування / дешифрування
1	СИМЕТРИЧНА КРИПТОСИСТЕМА	6	МАЄАИНШІОЦРЄХ	СЕКРЕТНИЙ
2	КРИПТОГРАФІЧНИЙ КЛЮЧ	3	РИІАККФДЖЯ	ШИФРУВАННЯ
3	СТІЙКІСТЬ ШИФРОСИСТЕМИ	7	ЦЛЩЯНЩДАЗЯАВК	ТАЄМНИЦЯ
4	ЦИКЛ ШИФРУВАННЯ	5	РИЯІКАИР	НАДІЙНІСТЬ

Продовження таблиці

Варіант	Фраза для шифрування	Зміщення k (для коду Цезаря)	Фраза для дешифрування	Ключ для шифрування / дешифрування
5	ГАМА-ПОСЛІДОВНІСТЬ	14	ДГЄДЖЮФШБІ	СТІЙКІСТЬ
6	АВТЕНТИЧНІСТЬ ДАНИХ І СИСТЕМ	9	ДМДСЮГ	ОБМЕЖЕННЯ
7	ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ	8	АВЕЩРОНАББЦД	ШПИГУН
8	ЕЛЕМЕНТАРНІ АЛГОРИТМИ ШИФРУВАННЯ	13	ЗФХГЙЮНААА	ЗАКРИТИЙ
9	ПЕРЕСТАНОВОЧНИЙ АЛГОРИТМ	9	МІЕІТУАШВ	ПРИХОВАТИ
10	ЛЕКСОГРАФІЧНИЙ ПОРЯДОК ПЕРЕСТАНОВОК	12	ШЮЗЮООЮНШУ	ДОСТУП
11	ІМОВІРНІСТЬ КРИПТОАНАЛІТИЧНОГО РОЗКРИТТЯ	2	ЦГЩАІАУОІЬБЧД	КОДУВАННЯ
12	ГЕНЕРАТОР ПСЕВДОВИПАДКОВИХ ЧИСЕЛ	10	ЯТДЬЬБШУЗ	ЗАХИСТ
13	ДЕШИФРУВАННЯ ВХІДНИХ ДАНИХ	11	СЕФЯЇГЙЛЇРАЯ	АВТЕНТИЧНИЙ

Контрольні запитання

1. Що вивчає наука криптографія?
2. Які існують основні типи криптографічних систем?
3. Як класифікуються методи кодування, в яких процес дешифрування є дзеркальним відображенням процесу шифрування?
4. Що таке криптоаналіз?
5. Яка математична закономірність покладена у реалізації методу XOR-шифрування?
6. У чому полягає метод шифрування Віженера?
7. Яка відмінність між шифром Віженера та шифром Бофорта?
8. Що собою являють блочні шифри?
9. Назвіть деякі приклади використання блочних шифрів?
10. Які історичні постатті користувалися шифрами? Якими саме?

Лабораторне заняття № 5

РЕАЛІЗАЦІЯ АЛГОРИТМІВ СИМЕТРИЧНОГО ШИФРУВАННЯ ДАНИХ

Мета: Ознайомитися із блоковими та поточними алгоритмами реалізації симетричних криптосистем

Необхідне програмне забезпечення

Компілятор мови програмування (на вибір викладача C++, C#, VB.NET, J# та ін.) із підтримкою роботи з програмним інтерфейсом Crypto API

Теоретична частина

Всі симетричні системи поділяються на дві великих групи:

- Блочні шифри;
- Потокові шифри.

Шифр називається ***блочним***, якщо шифрування кожного блоку відкритого тексту не залежить від інших блоків. Це означає, що результати двох однакових блоків одного і того ж відкритого тексту співпадають. Якщо ж результат шифрування чергового блоку відкритого тексту залежить не тільки від нього самого і секретного ключа, але й в загальному випадку від попередніх блоків, то цей шифр називають ***потоковим***. Існує більш м'яке «практичне» визначення: блочні шифри ті, в яких текст ділиться на порції по декілька октет, потокові ж системи оперують даними по одному біту або символу.

Також існує наступне формальне визначення системи блочного шифрування:

Шифр $F : P' K$ а C називається ***блочним***, якщо множини P, K, C – скінченні. В потоковому ж шифруванні ця вимога порушується. Наприклад, якщо на початку шифрування взагалі невідома кількість символів в тексті, то зручно вважати, що множина P не є скінченою. Відмітимо, що в більшості сучасних поточних системах результат шифрування поточного блоку відкритого тексту залежить від його номера і не залежить від самих попередніх блоків. Такі симетричні криптосистеми називають ***шифрами гаммування***. В даному випадку стійкість системи визначається виключно властивостями гамми (представляє собою псевдовипадкову числову послідовність).

Потокові шифри

В потокових шифрах, тобто при шифруванні потоку даних, кожен біт вхідної інформації шифрується незалежно від інших за допомогою гаммування.

В деяких потокових шифрах ключ коротший за повідомлення. Так, в системі Вернама для телеграфу використовується паперове кільце, яке містить гамму. Звичайно, стійкість такого шифру не ідеальна.

Зрозуміло, що обмін ключами розміром, що дорівнює розміру зашифрованої інформації не завжди доцільний. Тому частіше використовують гамму, яку отримують за допомогою генератора псевдовипадкових чисел (ПВЧ). В цьому випадку ключ – породжуючи число (початкове значення, вектор ініціалізації, initializing value, IV) для запуску генератора ПВЧ. Кожен генератор ПВЧ має період, після якого послідовність що генерується повторюється. Очевидно, що період псевдовипадкової гамми повинен перевищувати довжину зашифрованої інформації.

Генератор ПВЧ вважається коректним, якщо відслідковування фрагментів його виходу не дозволяє відновити пропущені частини або всю послідовність при відомому алгоритмі, але невідомому початковому значенні.

При використанні генератора ПВЧ можливі декілька варіантів:

1. Побітове шифрування потоку даних. Цифровий ключ використовується в якості початкового значення генератора ПВЧ, а вихідний потік бітів додається за модулем 2 з вихідною інформацією. В таких системах відсутня властивість розповсюдження помилок.

2. Побітове шифрування потоку даних зі зворотнім зв'язком (ЗЗ) по шифротексту. Така система аналогічна попередній, за виключенням того, що шифротекст повертається в якості параметра в генератор ПВЧ. Характерна властивість розповсюдження помилок. Область розповсюдження помилки залежить від структури генератора ПВЧ.

3. Побітове шифрування потоку даних із ЗЗ по вихідному тексту. Базою генератора ПВЧ є вихідна інформація. Характерною особливістю є можливість необмеженого розповсюдження помилки.

4. Побітове шифрування потоку даних із ЗЗ по шифротексту і по вихідному тексту.

Блочні шифри

При блочному шифруванні інформація розбивається на блоки фіксованої довжини і шифрується поблочно. Блочні шифри бувають двох основних видів:

- шифри перестановки (transposition, permutation, P-блоки);
- шифри заміни (подстановки, substitution, S-блоки).

Шифри перестановок переставляють елементи відкритих даних (біти, літери, символи) в деякому новому порядку. Розрізняють шифри горизонтальної, вертикальної, подвійної перестановки, решітки, лабіринти, лозунгові та інші.

Шифри заміни заміняють елементи відкритих даних на інші елементи за певним правилом. Розрізняють шифри простої, складної, парної заміни, літерно-складове шифрування і шифри колонної заміни. Шифри заміни діляться на дві групи:

- моноалфавітні (код Цезаря);
- поліалфавітні (шифр Відженера, циліндр Джефферсона, диск Уэтстоуна, шифрувальна машинка Enigma).

В сучасних криптографічних системах, як правило, використовують обидва способи шифрування (заміни і перестановки). Такий шифратор називають

складеним (product cipher). Він більш стійкий, ніж шифратор, що використовує тільки заміни або перестановки.

Блочне шифрування можна проводити двояко:

1. Без зворотнього зв'язку (ЗЗ). Декілька бітів (блок) вихідного тексту шифруються одночасно, і кожен біт вхідного тексту впливає на кожен біт шифротексту. Однак взаємного впливу блоків немає, тобто два однакових блока вихідного тексту будуть представлені однаковим шифротекстом. Тому подібні алгоритми можна використовувати тільки для шифрування випадкової послідовності бітів (наприклад, ключів). Прикладами є **DES** в режимі **ECB** та **ГОСТ 28147-89** в режимі простої заміни.

2. Зі зворотнім зв'язком. Зазвичай ЗЗ організовується так: попередній шифрований блок складається по модулю 2 з поточним блоком. В якості першого блоку в ланцюзі ЗЗ використовує ініціалізаційне значення. Помилка в одному біті впливає на два блоки – помилковий і наступний за ним. Приклад – **DES** в режимі **CBC**.

Генератор ПВЧ може застосовуватися при блочному шифруванні:

1. Поблочне шифрування потоку даних. Шифрування послідовних блоків (підстановки і перестановки) залежить від генератора ПВЧ, що управляється ключем.

2. Поблочне шифрування потоку даних із ЗЗ. Генератор ПВЧ управляється шифрованим або вихідним текстом або обома одразу.

Вельми розповсюдженим є федеральний стандарт США **DES** (Data Encryption Standard), на базі якого заснований міжнародний стандарт **ISO 8372-87**. DES був підтриманий Американським національним інститутом стандартів (American National Standards Institute, ANSI) і рекомендований для використання американською асоціацією банків (American Bankers Association, ABA).

DES передбачає 4 режими роботи:

- ECB (Electronic Codebook) електронний шифроблокнот;
- CBC (Cipher Block Chaining) ланцюжок блоків;
- CFB (Cipher Feedback) зворотній зв'язок по шифротексту;
- OFB (Output Feedback) зворотній зв'язок по виходу.

ГОСТ 28147-89 – російський стандарт шифрування даних. Стандарт включає три алгоритми шифрування (розшифровування) даних: режим простої заміни, режим гаммування, режим гаммування зі зворотнім зв'язком – режим вироблення імітовставки.

За допомогою імітовставки можна зафіксувати випадкову або навмисну модифікацію зашифрованої інформації. Виробляти імітовставку можна або перед зашифровуванням (після розшифровування) всього повідомлення або одночасно із зашифровуванням (розшифровуванням) по блокам. При цьому блок інформації шифрується першими шістнадцятьма циклами в режимі простої заміни, потім складається по модулю 2 із другим блоком, результат додавання знову шифрується першими шістнадцятьма циклами і т.д.

Алгоритми шифрування **ГОСТ 28147-89** наділені перевагами інших алгоритмів для симетричних систем і перевершують їх своїми можливостями. Так, **ГОСТ 28147-89** (256-битовый ключ, 32 циклу шифрування) в порівнянні з такими алгоритмами, як **DES** (56-бітовий ключ, 16 циклів шифрування) та **FEAL-1** (64-бітовий ключ, 4 цикли шифрування) володіє більш високою крипостійкістю за рахунок більш довшого ключа і більшого числа циклів шифрування.

Слід відмітити, що на відміну від **DES**, у **ГОСТ 28147-89** блок підстановки можна довільно змінювати, тобто він являється додатковим 512-бітовим ключем.

Алгоритми гаммування **ГОСТ 28147-89** (256-бітовий ключ, 512-бітовий блок підстановок, 64-бітовий вектор ініціалізації) перевершує по криптостійкості й алгоритм **B-Crypt** (56-бітовий ключ, 64-бітовий вектор ініціалізації). Перевагою **ГОСТ 28147-89** також є наявність захисту від нав'язування фальшивих даних (вироблення імітовставки) і однаковий цикл шифрування у всіх чотирьох алгоритмах **ГОСТ**у.

Блочні алгоритми можуть використовуватися і для вироблення гамми. В даному випадку гамма виробляється блоками і поблочно складається по модулю 2 з вихідним текстом. В якості прикладу можна назвати **B-Crypt**, **DES** в режимах **CFB** та **OFB**, **ГОСТ 28147-89** в режимах гаммування і гаммування із зворотнім зв'язком.

РЕАЛІЗАЦІЯ ПОТОКОВОГО АЛГОРИТМУ ГАММУВАННЯ

Гамма – це псевдо випадкова числова послідовність, що задається за певним алгоритмом і використовується для шифрування відкритих даних і розшифрування зашифрованих. Гаммуванням прийнято називати процес накладання за певним законом гамми шифру на відкриті дані для їх зашифрування.

Зазвичай використовується «виключаюче» АБО, яке також називається додаванням по модулю 2 і реалізоване в деяких мовах програмування оператором XOR. Для розшифрування та ж гамма накладається на зашифровані дані.

При однократному використанні випадкової гамми однакового розміру із шифрованими даними взлом коду неможливий (так звані криптосистеми з одноразовим або нескінченим ключем). В даному випадку «нескінчений» означає, що гамма не повторюється.

Для генерації дуже довгих ключів шифрування потрібно використати генератор псевдовипадкових чисел. Стандартні генератори типу функції *Random* / *Rnd* для шифрування використовувати не рекомендується, бо по-перше, вони легко піддаються криптоаналізу, по-друге, їх реалізації відрізняються в різних версіях компіляторів.

Для створення власного генератора можна використати дуже простий алгоритм лінійного конгруентного методу. Послідовність чисел в діапазоні від 0 до m-1 формується за рекурентною формулою

$$x_{k+1} = (a\, x_k + c) \bmod m$$

Д. Кнут в другому томі «Мистецтва програмування» наводить такі рекомендації для вибору парамерів:

1) x_0 -довільне;

2) a задовольняє таким умовам:

а) a – просте число;

б) a mod 8 =5 ;

в) $\sqrt{m} < a < m - \sqrt{m}$.

3) c – непарне число, таке що

$$\frac{c}{m} \approx \frac{1}{2} - \frac{1}{6}\sqrt{3} \approx 0.21132 .$$

Отримана послідовність псевдовипадкових чисел – гама шифру – накладається у вигляді послідовності байтів з допомогою операції XOR на файл, який шифрується. Пароль (ключ) використовується для вибору x_0.

Для 16-розрядних чисел можна вибрати:

Діапазон від 0 до 216-1 = 65535

c = 13849

$\sqrt{m} = 2^8 = 256$

$m - \sqrt{m} = 65536 - 256 = 65280$

$256 < a < 65280$

Блок гамування можна представити у вигляді наступної підпрограми:

```
Sub GammaCipher(OpenText, Gamma())
    Dim s, CryptText As String
    Dim i, k, k1 As Long
    Dim g, l As Integer
    Dim ch As Char
    k = gamma.Length
'Масив раніше згенерованих псевдовипадкових чисел
    s = OpenText
    CryptText = ""
    k1 = OpenText.TextLength
    For i = 1 To k1
       g = gamma(i Mod k)
'У випадку коли довжина гамми менша
'за довжину відкритого тексту,
'накладання гамми повторюється циклічно
       l = Asc(Mid(s, i, 1))
       ch = ChrW(l Xor g)
'Накладання байту гамми на
'байт відкритого тексту
       CryptText = CryptText + ch
'Формування рядка шифрованого тексту
    Next
End Sub
```

Реалізація блокового алгоритму Rijndael (AES) засобами програмного інтерфейсу CAPICOM (Crypto API)

Rijndael являється нетрадиційним блочним шифром, оскільки він виконаний в архітектурі SQUARE[1] . Алгоритм представляє кожен блок зашифрованих даних у вигляді двомірного масиву байт розміром 4x4, 4x6 або 4x8 в залежності від встановленої довжини блоку. Далі на відповідних етапах перетворення проводяться або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці. Авторами даного шифру є Джон Демен (Joan Daemen) та Вінсент Рамен (Vincent Rijmen).

Параметри шифру:

- розмір блоку 128, 192, 256 біт;
- розмір ключа 128, 192, 256 біт;
- число раундів 10, 12, 14. Залежить від розміру блоку (N_b) і ключа (N_k), заданих в бітах, за наступною формулою:

$$N_r = \frac{Max(N_b, N_k)}{32} + 6 .$$

Для реалізації даного криптоалгоритму скористаємося інструментами, які надає програмний інтерфейс Crypto API.

При симетричному шифруванні дані шифруються із використанням секретного значення. Для дешифрування потрібно те ж саме секретне значення (або ключ), що було використане для шифрування. Тому симетричне шифрування часто називають шифрування із спільним ключем (shared secret encryption).

Послуги симетричного шифрування надають декілька класів із простору імен System.Security.Cryptography. всі вони походять від класу System.Security.SymmetricAlgorithm і працюють майже однаково. Різниця між ними полягає в тому, що в кожному з класів реалізований свій алгоритм шифрування.

Алгоритм	Клас реалізації	Розмір ключа
DES	DESCryptoServiceProvider	64
TripleDES	TripleDESCryptoServiceProvider	192
RC2	RC2CryptoServiceProvider	128
Rijndael	RijndaelManaged	256

Клас симетричного шифрування простіше за все застосовувати з CryptoStream. CryptoStream – цей об'єкт-потік може служити оболонкою для будь-яких інших потоків, в тому числі для потоків, що являють собою файл, мережеве з'єднання, буфер пам'яті та інше. Для шифрування/дешифрування CryptoStream використовує клас алгоритму симетричного шифрування. Так, можна створити оболонку CryptoStream для потоку FileStream з метою прозорого шифрування даних по мірі їх запису або дешифрування по мірі зчитування.

[1] Архітектура побудови блочних шифрів із секретним ключем

```
Imports System.IO
Imports System.Security.Cryptography
_____________________________________________
Public Class Form1
  Private Rijndael As New RijndaelManaged

.......
End Class
```

При запуску програми створюється новий об'єкт RijndaelManaged, який інкапсулює данні ключа та вектора ініціалізації. Дані ключа зберігаються в файлі для подальшого використання:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    fKey = <Path_To_File_with_key> + "key.dat"
    If File.Exists(fKey) Then
       Dim fStream As FileStream = File.Open(fKey, FileMode.Open)
       Dim Key() As Byte
       ReDim Key(Rijndael.Key.Length - 1)
       Rijndael.Key = Key
       fStream.Close()
    Else 'Якщо ключа не знайдено, то створюємо
         'новий довільний ключ
       Dim fStream As New FileStream(fKey, FileMode.Create)
       fStream.Write(Rijndael.Key, 0, Rijndael.Key.Length)
       fStream.Close()
    End If
```

Алгоритми шифрування/дешифрування представимо у вигляді підпрограм:

Шифрування

```
Sub EnCrypting(FileName As String)
    Dim s As Char
    Dim Transform As ICryptoTransform = Rijndael.CreateEncryptor()
    Dim fStream As FileStream = File.Open(FileName,
                                          FileMode.OpenOrCreate)
    fStream.Write(Rijndael.IV, 0, Rijndael.IV.Length)

    Dim cStream As New
            CryptoStream(fStream,Transform,CryptoStreamMode.Write)
    Dim sWriter As New StreamWriter(cStream)

    sWriter.WriteLine(s)
'Записуємо зашифрований текст в файл
    sWriter.Flush()

' Об'єкт CryptoStream шифрує дані по одному блоку
' за раз. В кінці всієї
' операції ви повинні доповнити заключний частковий
' блок нулями і
' записати його в файл.
```

```
    cStream. FlushFinalBlock()
    sWriter.Close()
    cStream.Close()
    fStream.Close()
End Sub
```

Дешифрування

```
Sub DeCrypting(FileName As String)
      Dim s As String

      Dim fStream As FileStream = File.Open(FileName, FileMode.Open)
      Dim IV As Byte()
      ReDim IV(Rijndael.IV.Length - 1)
      fStream.Read(IV, 0, IV.Length)
      Rijndael.IV = IV

      Dim Transform As ICryptoTransform = Rijndael.CreateDecryptor()
'Створення криптографічного потоку в режимі читання, який буде
'декодувати двійкові дані відразу після їх зчитування з файлу

      Dim cStream As New
            CryptoStream(fStream,Transform,CryptoStreamMode.Read)
      Dim sReader As New StreamReader(cStream)
      s = sReader.ReadToEnd   'Розшифрований текст файлу
      sReader.Close()
      fStream.Close()

End Sub
```

Якщо потрібно отримати ключ на основі введеного пароля, можна використати наступний спосіб:

```
    ........
    Dim Salt(7) As Byte
    Dim random As New RNGCryptoServiceProvider()
        ' Використаємо генератор "надійних"
        ' псевдовипадкових чисел
    random.GetBytes(Salt)
    Dim Rijndael As New RijndaelManaged()
    Dim PDB As New PasswordDeriveBytes(TextPassword, Salt)
    Dim Key() As Byte
    Key = PDB.CryptDeriveKey("DES", "SHA", 64, Salt)
    ........
```

Практичне завдання

1. Реалізувати алгоритм шифрування текстового рядка гамуванням для стандартного генератора ПВЧ.
2. Використати генератор ПВЧ для шифрування бінарних файлів методом однопрохідного гамування.

3. Зашифрувати та розшифрувати текстовий файл за допомогою симетричного алгоритму шифрування Rijdael із використанням програмного інтерфейсу Crypto API.
4. Реалізувати пункт 3 для алгоритму DES та RC2 із генерацією ключа на основі введеного пароля.

Контрольні запитання

1. Яка основна відмінність між блоковими і потоковими симетричними криптосистемами?
2. Що таке «надійне» випадкове число?
3. Чим псевдовипадкові послідовності відрізняються від випадкових?
4. Який генератор ПВЧ називається коректним?
5. Що таке гамма?
6. В чому полягає алгоритм гамування?
7. Чому для генерування секретних ключів та гамми не рекомендується користуватися стандартними генераторами ПВЧ типу Random/Rnd?
8. Що таке «чисті» генератори випадкових чисел?
9. Яка операція використовується при реалізації гаммування на основі мікропроцесорної техніки?
10. Назвіть найвідоміші блочні криптосистеми.

Лабораторне заняття № 6

CRYPTO API. АСИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ ДАНИХ

Мета: Розглянути можливості та особливості використання програмного інтерфейсу Crypto API та реалізувати деякі асиметричні алгоритми шифрування даних за його допомогою.

Необхідне програмне забезпечення

Компілятор мови програмування (на вибір викладача C++, C#, VB.NET, J# та ін.) із підтримкою роботи з програмним інтерфейсом Crypto API

Теоретична частина

ОГЛЯД МОЖЛИВОСТЕЙ ПРОГРАМНОГО ІНТЕРФЕЙСУ CAPICOM

Швидкий розвиток електронної комерції різко підсилив потребу в надійних алгоритмах шифрування для захисту конфіденційної інформації. Для вирішення даного завдання Microsoft ще у 1996 р. ввела **Cryptography API** (Crypto API). Еволюція засобів захисту сприяє появі нових криптографічних сервісів, таких як простір імен **System.Security.Cryptography** загальномовного виконуючого середовища (common language runtime, CLR) в **Microsoft .NET**.

Це простір імен відкриває програмний доступ до найрізноманітніших криптографічних сервісів, за допомогою яких додатки можуть шифрувати і дешифрувати дані, забезпечувати їх цілісність, а також обробляти цифрові підписи і сертифікати. Деякі з таких сервісів застосовуються безпосередньо в самій **.NET Framework** (наприклад для підтримки аутентифікації в **ASP.NET**).

Простір імен Cryptography

На найвищому рівні простір імен **Cryptography** можна розбити на чотири основні частини (табл. 1). Головне призначення цього простору – надавати класи з алгоритмами таких операцій, як шифрування і створення хешів. Ці алгоритми реалізуються на основі розширюваного шаблону (pattern), що включає два рівні наслідування.

На вершині ієрархії розташовується абстрактний базовий клас (ніби **AsymmetricAlgorithm** або **Hash Algorithm**), ім'я якого відповідає типу алгоритму. Від такого класу успадковує абстрактний клас другого рівня, що надає відкритий інтерфейс для використання даного алгоритму. Наприклад, **SHA1** (Secure Hash Algorithm) є похідним від **HashAlgorithm** клас і містить методи і властивості, специфічні для алгоритму **SHA1**. Нарешті, сама реалізація алгоритму є похідною від класу другого рівня; саме її екземпляр створюється і використовується клієнтським додатком. На цьому рівні реалізація може бути керованою, некерованою або і тій й іншій.

Таблиця 1

Основні елементи простору імен Cryptography

Елемент	Опис
Алгоритми шифрування	Набір класів, вживаних для реалізації алгоритмів симетричного і асиметричного шифрування, а також хешування
Допоміжні класи	Класи, що забезпечують генерацію випадкових чисел, виконання перетворень, взаємодія із простором розміщення Crypto API й саме шифрування на основі потокової моделі
Сертифікати X.509	Класи, що визначені в просторі імен System.Security. Cryptography.X509 Certificates і які надають цифрові сертифікати
Цифрові підписи XML	Класи, визначені в просторі імен System.Cryptography.Xml і представляючі цифрові підписи в XML-документах

До імен некерованих реалізацій зазвичай додається суфікс «**CryptoServiceProvider**» (скажімо, SHA1CryptoServiceProvider), вказуючий на те, що дана реалізація насправді надається провайдером криптографічного сервісу (Cryptographic Service Provider, CSP), який встановлений на рівні операційної системи і діє як оболонка **Crypto API**. У імена керованих реалізацій включається суфікс «**Managed**» (наприклад SHA1Managed). Такі реалізації не спираються на **Crypto API** і містять виключно керований код.

Ще один важливий момент. При створенні екземпляра одного з конкретних класів початкові конструктори завжди записують в параметри об'єкту розумні і безпечні значення (якщо це можливо). Так, **алгоритми асиметричного шифрування**, що спираються на криптографію з відкритим ключем, генерують **випадкову пару ключів**, а **алгоритми симетричного шифрування** – **випадковий ключ** і **вектор ініціалізації** (initialization vector, IV); при цьому вони автоматично налаштовують такі властивості, як **Mode** і **Padding**. Більш того, алгоритми другого типу за умовчанням прагнуть використовувати «стійкі» значення.

Другий основний набір класів в просторі імен **System.Security. Cryptography** включає як класи, реально вживані в процесі шифрування і розшифровки даних, так і різноманітні допоміжні класи. Це простір імен містить, наприклад, абстрактний клас **RandomNumberGenerator**, від якого успадковують класи **RNGCryptoServiceProvider**, **ToBase64Transform** і **FromBase64Transform** (використовуються при відповідних перетвореннях даних).

Простір імен **Cryptography** не тільки надає алгоритми шифрування, але і містить дочірній простір імен **X509Certificates**.

Останнє об'єднує всього три класи, призначених для операцій з сертифікатами **Authenticode X.509 v.3.** Клас **X509Certificate** надає статичні методи **CreateFromCertFile** і **CreateFromSignedFile** для створення екземпляра сертифікату:

```
Dim c As X509Certificate
c = X509Certificate.CreateFromCertFile("myCert.cer")
Console.WriteLine(c.GetName)
Console.WriteLine(c.GetPublicKeyString)
Console.WriteLine(c.GetSerialNumberString)
Console.WriteLine(c.GetExpirationDateString)
```

Після цього сертифікат можна використовувати в декількох цілях, зокрема для перевірки на Web-сервері: ви пов'язуєте сертифікат з клієнтським запитом через властивість **ClientCertificates** об'єкту **System.Net.HttpWebRequest** (воно відкриває доступ до об'єкту **X509CertificateCollection**).

Простір імен **Cryptography**, зокрема, відкриває доступ до алгоритмів симетричного шифрування (далі – симетричні алгоритми).

Симетричні алгоритми називаються так тому, що для шифрування і дешифровки використовується один секретний ключ. Очевидно, що і відправник, і одержувач повинні тримати такий ключ в секреті, інакше шифрування стане безглуздим. Крім того, в режимі **CBC** симетричні алгоритми використовують **IV** – несекретне двійкове значення; воно ініціалізувало алгоритм і вносить додаткові варіації до процесу шифрування. **SymmetricAlgorithm** є абстрактним базовим класом, від якого успадковують інші класи, специфічні для конкретних алгоритмів.

До підтримуваних симетричних алгоритмів належать **Data Encryption Standard** (DES), **RC2**, **Rijndael** і **Triple Data Encryption Standard** (TRIPLEDES). Кожен алгоритм включає який-небудь похідний від **SymmetricAlgorithm** абстрактний клас ніби **DES** і похідний від базового керований клас або клас провайдера сервісу, наприклад **DESCryptoServiceProvider**.

Другий тип – асиметричне шифрування або шифрування з відкритим ключем (далі – асиметричні алгоритми); відповідний клас є похідним від абстрактного класу **AsymmetricAlgorithm**.

До загальновідомих асиметричних алгоритмів відносяться **Digital Signature Algorithm** (DSA) і **RSA**. У асиметричних алгоритмах застосовується пара ключів: один – закритий, інший – відкритий. Як правило, відкритий ключ доступний кому завгодно і використовується відправником для шифрування даних, тоді як закритий ключ зберігається в захищеному місці і застосовується для розшифровки даних, зашифрованих за допомогою відкритого ключа. Найбільш поширений алгоритм RSA.

Ці алгоритми кінець кінцем є похідними від абстрактних класів **DSA** і **RSA**, у свою чергу похідних від **AsymmetricAlgorithm**. Оскільки ці алгоритми дуже складні, їм потрібні допоміжні класи, похідні, наприклад, від **AsymmetricKeyExchangeFormatter** і **AsyrametricSignatureFormatter**

АСИМЕТРИЧНІ АЛГОРИТМИ ШИФРУВАННЯ

В асиметричних алгоритмах шифрування (або криптографії з відкритим ключем) для зашифровування інформації використовують один ключ (відкритий), а для розшифровки – інший (секретний). Ці ключі різні і не можуть бути отримані один з іншого.

Схема обміну інформацією така:

- одержувач обчислює відкритий і секретний ключі, секретний ключ зберігає в таємниці, відкритий же робить доступним (повідомляє відправника, групу користувачів мережі, публікує);
- відправник, використовуючи відкритий ключ одержувача, зашифровує повідомлення, яке пересилається одержувачеві;
- одержувач отримує повідомлення і розшифровує його, використовуючи свій секретний ключ.

Всі криптосистеми з відкритим ключем досить повільні, і жодна з них не може зрівнятися по швидкодії з симетричними криптосистемами. Так швидкодія **RSA** в тисячі разів нижча, ніж у **DES** або **ГОСТУ 28147-89**. Тому використовують гібридні криптосистеми.

- На початковому етапі учасники інформаційного обміну, використовуючи протокол вироблення загального секретного ключа, формують загальну секретну інформацію (сеансовий ключ). На наступному етапі обміну використовується криптосистема з секретним ключем.
- Відправник виробляє секретний ключ – випадкове число, використовуване тільки один раз і тому зване одноразовим або сеансовим ключем. Цей ключ використовується для шифровки повідомлення за допомогою симетричного алгоритму. Сеансовий ключ зашифровується на відкритому ключі одержувача і приєднується до раніше зашифрованого документу. Сформоване таки образом повідомлення відсилається одержувачеві. Останній, отримавши повідомлення, повторює ті ж процедури, але в зворотньому порядку – за допомогою свого секретного ключа він відновлює сеансовий ключ і розшифровує повідомлення.

Криптосистема **RSA** (захищена патентом США N 4405829) була розроблена в 1977 році в Массачусетському технологічному інституті й отримала назву на честь її творців: **Рона Рівеста** (Ron Rivest), **Аді Шаміра** (Adi Shamir) і **Леонарда Адлмана** (Leonard Adleman). Вони скористалися тим фактом, що знаходження великих простих чисел в обчислювальному відношенні здійснюється легко, але розкладання на множники добутку двох таких чисел практично нездійсненно. Доведено (теорема Рабіна), що розкриття шифру **RSA** еквівалентно такому розкладанню. Тому для будь-якої довжини ключа можна дати нижню оцінку числа операцій для розкриття шифру, а з урахуванням продуктивності сучасних комп'ютерів оцінити і необхідне на цей час. Можливість гарантовано оцінити захищеність алгоритму **RSA** стала однією з причин популярності цієї криптосистеми на фоні десятків інших схем. Тому алгоритм **RSA** використовується в банківських комп'ютерних мережах, особливо для роботи з віддаленими клієнтами (наприклад, при обслуговуванні операцій з кредитними картками).

Алгоритм шифрування RSA можна коротко описати наступним чином:

1. Випадковим чином вибираються два дуже великих простих числа p та q.
2. Обчислюються два добутки $n - p \times q$ та $n - (p - 1) \times (q - 1)$.

3. Вибирається випадкове ціле число E, яке не має спільних множників із n.
4. Вибирається D, таке, що $DE \sim l$ по модулю n.
5. Вихідний текст X, розбивається на блоки таким чином, що $0 < X < n$.
6. Для шифрування повідомлення необхідно обчислити $C - X^E$ по модулю n.
7. Для дешифрування обчислюється $X - C$ по модулю n.

Таким чином, щоб зашифрувати повідомлення необхідно знати пару чисел (E, n), а щоб дешифрувати – пару чисел (D, n). Перша пара – це відкритий ключ, а друга – закритий. Знаючи відкритий ключ (E, n), можна обчислити значення закритого ключа D. Необхідною проміжною дією в цьому перетворенні є знаходження чисел p і q, для чого потрібно розкласти на прості множники дуже велике число n, а на це необхідно витратити багато часу. Саме на основі даного факту базується *криптостійкість* алгоритму RSA. В окремих джерелах приводяться наступні оцінки: для того, щоб знайти розклад 200-значного числа, знадобиться 4 мільярда років роботи комп'ютера із швидкодією мільйон операцій в секунду. Однак варто врахувати те, що зараз активно ведуться роботи по вдосконаленню методів розкладу великих чисел, тому в алгоритмі RSA намагаються використовувати числа довжиною більше 200 десяткових розрядів.

На думку фахівців, RSA є одним з найбільш розвинених методів криптографічного захисту інформації з відкритим ключем, на основі якого організуються найефективніші і перспективні системи захисту даних. В таких системах для шифрування даних використовується один ключ, а для розшифрування – інший, тобто створюється ключова пара. Причому така пара формується одержувачем. Перший ключ не є секретним і може бути опублікований для використання всіма користувачами системи, що шифрують дані, або ж може бути взятим з вже одержаного документу. Розшифрування даних за допомогою відомого ключа неможливе, тому що для цих цілей їх одержувач використовує інший ключ, який є секретним. Природно, ключ розшифрування при цьому не може бути визначений з ключа шифрування.

Внаслідок складності реалізації операцій модульної арифметики криптоалгоритм RSA часто використовують тільки для шифрування невеликих об'ємів інформації, наприклад для розсилки класичних секретних ключів. Програмна реалізація криптоалгоритмів типу RSA значно складніша й менш продуктивніша, ніж реалізація класичних криптоалгоритмів типу DES.

Ефективним програмно-технічним засобом для запровадження криптосистем є застосування електронних ключів. Одним з перших електронних ключів була розроблена в кінці 80-х років спеціалізована інтегральна схема IDS-P2 (MB8763). Пізніше програмна реалізація алгоритму RSA здійснена на інтегральній схемі CY1024.

З останніх слід відзначити розробки фірми MOTOROLA, яка випускає швидкодіючих криптографічних мікросхем SC49A, що є 8-бітний мікроконтролером з 1024-біт сопроцесором, 20-кбіт ROM, 4-кбіт EEPROM, 896-байт RAM і робочою напругою 3-5 вольт.

РЕАЛІЗАЦІЯ АСИМЕТРИЧНОГО АЛГОРИТМУ RSA ЗАСОБАМИ ПРОГРАМНОГО ІНТЕРФЕЙСУ CRYPTO API

Асиметричне шифрування дозволяє двом сторонам обмінюватися зашифрованими даними, не застосовуючи секретне значення. З асиметричним шифруванням тісно пов'язано поняття пари ключів. Суть асиметричного шифрування в тому, що кожен користувач має закритий і відкритий ключі, при цьому інформація, зашифрована відкритим ключем, може бути розшифрована тільки закритим ключем. Відкритий ключ робиться доступним всьому світу, і його можна вільно передавати по незахищених з'єднаннях, скажімо, по Інтернету. Закритий ключ тримається в суворому секреті. Таким чином, зашифрувати повідомлення відкритим ключем одержувача даних може будь-який користувач, але єдиною людиною, яка зможе розшифрувати таке повідомлення, є згаданий цільовий одержувач повідомлення, у якого є відповідний закритий ключ.

Для асиметричного шифрування даних в **.NET** служить клас **RSACryptoServiceProvider**, що підтримує ключі розміром 384-16384 біт (із збільшенням по 8 біт). За умовчанням розмір ключа рівний 1024 бітам. Як правило, чим більше розмір ключа, тим надійніше шифрування, проте надійність асиметричних і симетричних ключів різна. За деякими оцінками надійність 1024-бітового ключа **RSA** приблизно еквівалентна надійності 75-бітового симетричного ключа.

Асиметричне шифрування значно незручніше симетричного. Воно не використовує потокову модель і змушує шифрувати дані невеликими блоками. Якщо ви спробуєте зашифрувати дані, що перевищують за об'ємом один блок, не розбивши їх, відбудеться помилка. Для вирішення цієї проблеми ви можете скомбінувати асиметричне шифрування з симетричним.

Асиметричне шифрування зазвичай працює в 1000 разів повільніше, аніж симетричне; крім того, об'єм зашифрованих по асиметричному алгоритму даних у декілька разів перевищує об'єм даних, зашифрованих симетрично. Ви можете уникнути цих обмежень, згенерувавши випадковий симетричний ключ і зашифрувавши блок своїх даних з його допомогою. Хитрість в тому, що після цього вам слід асиметрично зашифрувати випадковий ключ, використовуючи відкритий ключ одержувача, і додати зашифрований симетричний ключ в зашифрований документ. Одержувач зможе витягнути зашифрований симетричний ключ, розшифрувати його своїм закритим ключем і за допомогою симетричного ключа розшифрувати частину документа, що залишилася. Цей спосіб застосовується досить часто; як приклади можна привести протокол **SSL** (при використанні якого для кожної взаємодії передається симетричний ключ сеансу) і файлову систему **Windows Encrypting File System** (при цьому випадковий симетричний ключ генерується для кожного шифрованого файлу).

```
.....
    Inherits System.Windows.Forms.Form
  Private RSA As New Security.Cryptography.RSACryptoServiceProvider()

Private Sub Form_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

```
' Перевірка наявності файлу ключів.
' Цей файл міститиме і відкритий, і закритий ключі.
    If File.Exists("key.bin") Then
        Dim fs As New IO.FileStream("key.bin", FileMode.Open)
        Dim r As New IO.StreamReader(fs)
        RSA.FromXmlString(r.ReadToEnd())
        fs.Close()
    Else
        Dim fs As New FileStream("key.bin", FileMode.CreateNew)
        Dim w As New StreamWriter(fs)
        w.Write(RSA.ToXmlString(True))
        w.Flush()
        fs.Close()
    End If
End Sub
```

Шифрування

```
Private Sub Crypting()
    txtsource = "Some text source."

' Відкриття файлу для запису.
    Dim fs As New FileStream("testfile.bin", FileMode.Create)

' Створення нового (випадкового) симетричного ключа.
    Dim Rijndael As New RijndaelManaged

' Шифрування симетричного ключа і вектора ініціалізації
' з використанням відкритого ключа RSA.
    Dim EncryptedKey() As Byte = RSA.Encrypt(Rijndael.Key, False)
    Dim ENCRYPTEDLV() As Byte = RSA.Encrypt(Rijndael.IV, False)

' Запис асиметрично зашифрованих ключа
' і вектора ініціалізації у файл.
    fs.Write(EncryptedKey, 0, EncryptedKey.Length)
    fs.Write(ENCRYPTEDIV, 0, ENCRYPTEDLV.Length)

' Запис частини файлу, що залишилася, з використанням симетричного
' шифрування.
    Dim Transform As ICryptoTransform = Rijndael.CreateEncryptor()
    Dim cs As New CryptoStream(fs, Transform, CryptoStreamMode.Write)
    Dim w As New StreamWriter(cs)
    w.Write(txtsource)
    w.Flush()
    cs.Close()
    w.Close()
    fs.Close()
End Sub
```

Дешифрування

```
Private Sub Decrypting()
' Відкриття файлу для читання.
    Dim fs As New FileStream("testfile.bin", FileMode.Open)
```

```
' Розмір ключа вимірюється в бітах. 8 біт рівні 1 байту.
' Число байт в зашифрованому блоці
' даних завжди рівно розміру ключа.
  Dim EncryptedBlockSize As Integer = CType(RSA.KeySize / 8, Integer)

' Отримання зашифрованого ключа і вектора ініціалізації.
    Dim Rijndael As New RijndaelManaged
    Dim EncryptedKey(EncryptedBlockSize - 1) As Byte
    Dim ENCRYPTEDIV(EncryptedBlockSize - 1) As Byte
    fs.Read(EncryptedKey, 0, EncryptedKey.Length)
    fs.Read(ENCRYPTEDIV, 0, ENCRYPTEDIV.Length)
    Rijndael.KeySize = EncryptedBlockSize
    Rijndael.Key = RSA.Decrypt(EncryptedKey, False)
    Rijndael.IV = RSA.Decrypt(ENCRYPTEDIV, False)

' Частина файлу, що залишилася, прочитується
' за допомогою симетричного ключа.
    Dim Transform As ICryptoTransform = Rijndael.CreateDecryptor()
    Dim cs As New CryptoStream(fs, Transform, CryptoStreamMode.Read)
    Dim r As New StreamReader(cs)

' Запис дешифрованих даних
    txtsource = r.ReadToEnd()
    r.Close()

End Sub
```

Практичне завдання

1. Отримати список всіх встановлених в системі криптопровайдерів.
2. Виконати асиметричне шифрування даних за допомогою класу RSACryptoServiceProvider та симетричного криптопровайдера RijndaelManaged.
3. Побудувати для малих n, D та E криптосистему RSA (E, n).

Контрольні запитання

1. Що собою представляє програмний інтерфейс Crypto API?
2. Які криптоалгоритми називаються асиметричними?
3. Які криптоалгоритми підтримуються у Crypto API?
4. Які переваги асиметричних алгоритмів?
5. Які недоліки асиметричних алгоритмів?
6. Що таке секретний (відкритий) ключ?
7. Де найчастіше використовуються асиметричні алгоритми?
8. У якому вигляді найчастіше реалізуються асиметричні алгоритми і чому?
9. Який математичний апарат лежить в основі побудови криптосистеми RSA?
10. Чому недоцільно використовувати асиметричні криптосистеми при шифруванні великих повідомлень, а лише у поєднанні із симетричними?

Лабораторне заняття № 7

ЗАХИСТ ДАНИХ ВІД МОДИФІКАЦІЙ ТА ПІДМІНИ

Мета: Ознайомитися із криптографічними методами захисту даних від модифікації та підміни

Необхідне програмне забезпечення

Компілятор мови програмування (на вибір викладача C++, C#, VB.NET, J# та ін.) із підтримкою роботи з програмним інтерфейсом Crypto API

Теоретична частина

ЦИФРОВИЙ ПІДПИС

При передачі інформації повинні бути забезпечені разом або окремо:

1. Конфіденційність (privacy) – зловмисник не повинен мати можливості дізнатися зміст переданого повідомлення.

2. Достовірність (authenticity), яка включає два поняття

- цілісність (integrity) – повідомлення повинне бути захищене від випадкової або умисної зміни;
- ідентифікація відправника (перевірка авторства) – одержувач повинен мати можливість перевірити, ким відправлено повідомлення.

Шифрування може забезпечити конфіденційність, а в деяких системах і цілісність.

Цілісність повідомлення перевіряється обчисленням контрольної функції (check function) від повідомлення – якогось числа невеликої довжини. Ця контрольна функція повинна з високою вірогідністю змінюватися навіть при малих змінах повідомлення (видалення, включення, перестановки або переупорядковування інформації). Називають і обчислюють контрольну функцію порізному:

- код достовірності повідомлення (Message Authentical Code, MAC);
- квадратичний конгруентний алгоритм (Quadratic Congruentical Manipulation Detection Code, QCMDC);
- Manipulation Detection Code (MDC);
- Message Digest Algorithm (MD5);
- контрольна сума;
- символ контролю блоку (Block Check Character, BCC);
- циклічний надмірний код (ЦВК, Cyclic Redundancy Check, CRC);
- хеш-функція (hash);
- імітовставка в ГОСТ 28147-89;
- алгоритм з усіканням до n бітів (n-bit Algorithm with Truncation).

При обчисленні контрольної функції може використовуватися який-небудь алгоритм шифрування. Можливе шифрування самої контрольної суми.

Широко застосовується **цифровий підпис** (цифрове доповнення до інформації, що передається, яке гарантує цілісність останньої і дозволяє перевірити її авторство).

Відомі моделі **цифрового підпису** (digital signature) на основі алгоритмів симетричного шифрування, але при використанні систем з відкритими ключами цифровий підпис здійснюється зручніше.

Для використання алгоритму **RSA** повідомлення слід стиснути функцією хешування (алгоритм **MD5** – Message Digest Algorithm) до 256-бітового хешу *(H)*. Сигнатура повідомлення S обчислюється таким чином:

$$S = H^{d} \bmod n$$

Сигнатура пересилається разом з повідомленням.

Процес ідентифікації полягає в отриманні хеш-функції повідомлення *(H')* і порівнянні з

$$H = S^{e} \bmod n$$

де *H* – хеш повідомлення

S – його сигнатура

d – секретний ключ,

e – відкритий ключ.

Перевірці достовірності присвячені стандарти:

- перевірка достовірності (аутентифікація, authentication) – ISO 8730-90, ISO/IES 9594-90 та ITU X.509[2] ;
- цілісність – ГОСТ 28147-89, ISO 8731-90;
- цифровий підпис – ISO 7498, P 34.10-94 (Росія), DSS (Digital Signature Standard, США).

При розробці механізму цифрового підпису виникають наступні задачі:

- створити підпис таким чином, щоб її неможливо було підробити;
- мати можливість перевірки того, що підпис дійсно належить вказаному власнику;
- мати можливість запобігти відмові від підпису.

Класична схема створення цифрового підпису

При створенні цифрового підпису за класичною схемою відправник

- застосовує до вихідного повідомлення хеш-функцію;
- обчислює цифровий підпис за хеш-образом повідомлення з використанням секретного ключа створення підпису;
- формує нове повідомлення, яке складається з вихідного повідомлення та доданого до нього цифрового підпису.

[2] **ISO** – Міжнародна організація по стандартизації (МОС), **ITU** – Міжнародний союз електрозв'язку (МСЕ).

Отримувач, відкривши підписане повідомлення,

- відділяє цифровий підпис від основного повідомлення;
- застосовує до основного повідомлення хеш-функцію;
- з використанням відкритого ключа перевірки підпису здобуває хеш-образ повідомлення з цифрового підпису;
- перевіряє відповідність обчисленого хеш-образу повідомлення (п.2) та добутого з цифрового підпису. Якщо хеш-образи співпадають, то підпис визнається справжнім.

Схема підпису RSA

Криптосистема з відкритим ключем RSA може використовуватись не лише для шифрування, але і для побудови схеми цифрового підпису.

Для створення підпису повідомлення M відправник

- обчислює хеш-образ r = h(M) повідомлення M за допомогою деякої хеш-функції;
- зашифровує отриманий хеш-образ *r* на своєму секретному ключі *(d,n)*, тобто обчислює значення $s = r^d \bmod n$, що і є підписом.
- Для перевірки підпису отримувач
- розшифровує підпис s на відкритому ключі *(e,n)* відправника, тобто обчислює $r' = s^e \bmod n$ і таким чином відновлює імовірний хеш-образ *r'* повідомлення *M*;
- обчислює хеш-образ *h(M) = r* повідомлення M за допомогою тієї ж самої хеш-функції, котру використовував відправник;
- порівнює отримані значення *r* і *r'* . Якщо вони співпадають, то підпис правильний, відправник дійсно є тим, за кого себе видає, і повідомлення не було змінено при передачі.

ХЕШ-ФУНКЦІЯ

На відміну від рукописного тексту, електронний документ дуже легко може піддатися різним, непомітним для одержувача документа, змінам. Тому часто виявляється необхідним не зашифровувати вміст інформаційного повідомлення, а забезпечити гарантії достовірності авторства і визначити, чи не вносилися несанкціоновані автором зміни в інформацію. Для цих цілей використовується алгоритм цифрового підпису. При використанні цифрового підпису інформація не шифрується і залишається доступною будь-якому користувачеві, що має до неї доступ. Для визначення достовірності автора і вмісту використовується спеціальна функція, звана “дайджест” або хеш-функція.

Процес підпису документа виглядає таким чином. На першому кроці будується спеціальна функція, що нагадує контрольну суму, – хеш-функція, вона ідентифікує вміст документа. На другому кроці автор документа шифрує вміст хеш-функції своїм персональним закритим ключем. Зашифрована хеш-функція поміщається в те ж повідомлення, що і сам документ. Отримане повідомлення може зберігатися на будь-якому носієві або пересилатися по електронній пошті. Хеш-функція має невеликий розмір і збільшує розмір повідомлення ненабагато.

При прочитанні оснащуваного електронним підписом документу користувач може переконається в його достовірності. Алгоритм верифікації електронного підпису полягає в наступному. На першому етапі одержувач повідомлення будує власний варіант хеш-функції підписаного документа. На другому етапі відбувається розшифровка хеш-функції, що міститься в повідомленні. На третьому етапі відбувається порівняння двох хеш-функцій. Їх збіг гарантує одночасно достовірність вмісту документа і його авторства.

При пересилці документа по електронній пошті відкритий ключ автора може міститися в тому ж повідомленні, що і сам документ, це спрощує процес верифікації підпису.

Функцією хешування (хеш-функцією або вкорочуючою функцією) називається перетворення даних, що переводить рядок бітів *M* довільної довжини в рядок бітів *h(M)* деякої фіксованої довжини (кілька десятків чи сотень біт).

Хеш-функція *h(M)* повинна задовольняти наступним умовам:

- хеш-функція *h(M)* повинна бути чутливою до будь-яких змін вхідної послідовності *M*;
- для даного значення *h(M)* повинно бути неможливо знайти значення *M*;
- для даного значення *h(M)* повинно бути неможливо знайти значення $M' \neq M$ таке, *що* $h(M') = h(M)$.

Ситуація, за якої для різних вхідних послідовностей M , M' співпадають значення їх хеш-образів: $h(M) = h(M')$, називається колізією.

При побудові хеш-образу, вхідна послідовність M розбивається на блоки M_i фіксованої довжини і оброблюється поблочно за формулою

$$H_i = f(H_{i-1} , M_i).$$

Хеш-значення, що обчислюється при введенні останнього блоку повідомлення, стає хеш-значенням всього повідомлення.

В якості прикладу розглянемо спрощений варіант хеш-функції із рекомендацій МККТТ X.509:

$$H_i = (H_{i-1} + M_i)^2 \bmod n ,$$

де $n = pq$, p і q – великі прості числа, H_0 – довільний початковий вміст, M_i – і-тий блок повідомлення $M = M_1 M_2 \ldots M_k$.

ВИКОРИСТАННЯ ХЕШ-ФУНКЦІЇ ДЛЯ ПЕРЕВІРКИ НЕЗМІННОСТІ ДАНИХ

Шифрування не дозволяє зловмисникам прочитати ваші дані, але воно не захищає дані від підміни. Для цього потрібний деякий спосіб перевірки даних на предмет того, що вони не були змінені. Цю роль і виконують хеш-коди.

Алгоритм обчислення хеш-коду генерує невеликий (зазвичай близько 20 байт) двійковий відбиток будь-яких даних, які можуть бути представлені у формі послідовності байт. Хеш-коди криптографічно-безпечні: інакше кажучи, зловмисникові буде украй складно створити документ, якому відповідатиме конкретний хеш-код. Крім того, навіть дізнавшись хеш-код, зломщик не отримає інформації про початкові дані. І хоча різним даним можуть відповідати однакові хеш-коди, статистично це маловірогідно. Фактично навіть при

щонайменшій зміні даних (наприклад, при зміні 1 бита) є 50%-а вірогідність незалежної зміни кожного біта хеш-коду.

У таблиці 1 наведений список алгоритмів обчислення хеш-коду, представлених класами з простору імен **System.Security.Cryptography**. Всі ці класи проведені від класу **HashAlgorithm** і включають метод **ComputeHash**, який приймає масив байт або потік і повертає масив байт, що містить хеш-код. Як і у разі алгоритмів шифрування, чим менше розмір хешу, тим слабший забезпечуваний ним захист (і тим легше зломщикові знайти інший набір даних, якому відповідає той же хеш-код).

Таблиця 1

Класи алгоритмів генерування хешу

Алгоритм	Клас реалізації за умовчанням	Розмір хешу (у бітах)
MD5	MD5CryptoServiceProvider	128
SHA-1	SHA1CryptoServiceProvider	160
SHA-256	SHA256Managed	256
SHA-384	SHA384Managed	384
SHA-512	SHA512Managed	512

Наступний фрагмент підраховує хеш-коди для тестового файлу до і після невеликої зміни і порівнює їх. Він включає допоміжну функцію, що перевіряє масиви байт на рівність шляхом перебору всіх байт:

```
Private Function CompareByteArray(ByVal BYTESA() As Byte, ByVal BYTESB()
       As Byte) As Boolean

    If Not BYTESA.Length = BYTESB.Length Then Return False
    Dim i As Integer
    For i = 0 To BYTESA.Length - 1
        If Not BYTESA(i) = BYTESB(i) Then Return False
    Next
    Return True

End Function

Private Sub Hashing()

' Створення нового файлу.
    Dim fs As New FileStream("testfile.bin", FileMode.Create)

' Запис деяких даних.
    Dim w As New StreamWriter(fs)
    w.WriteLine("This is the first line.")
    w.WriteLine("This is the second line.")
    w.Flush()

' Обчислення 512-бітового (64-байтового) хеш-коду для файлу.
    Dim SHA As New SHA512Managed()
    fs.Position = 0
    Dim HASHA() As Byte = SHA.ComputeHash(fs)
```

```
' Відображення хеш-коду.
    TextSource1 = BitConverter.ToString(HASHA)

' Додавання символу у файл і повторне обчислення хеш-коду.
    w.Write("!")
    w.Flush()
    fs.Position = 0
    Dim HASHB() As Byte = SHA.ComputeHash(fs)

    If CompareByteArray(HASHA, HASHB) Then
        MessageText = "Хеш-коди ідентичні!"
    Else
        MessageText = "Невідповідність хеш-кодів!"
    End If

    TextSource2 = BitConverter.ToString(HASHB)
    fs.Close()

End Sub
```

Для гарантії цілісності даних зберігайте хеш-код в безпечному місці (такому, наприклад, як захищена БД), інакше зловмисник зможе підробити документ, згенерувати новий хеш-код і замінити ним оригінальний хеш-код. Якщо хеш-код потрібно зберігати в небезпечному місці або разом з самими даними, створіть хеш-код, захищений від підміни.

Створення хеш-коду, захищеного від підміни

Хеш-коди мають одне очевидне обмеження: якщо не зберігати їх в безпечному місці, ніщо не перешкодить зловмисникові підмінити ваші дані і згенерувати новий хеш-код, відповідний зміненим даним. Таку зміну даних визначити неможливо. Ця проблема часто зустрічається при розробці розподілених додатків, що використовують хеш-коди для перевірки повідомлень отримуваних по мережі.

Вирішенням даної проблеми є створення хеш-коду, що не допускає відтворення зловмисником. Наприклад, ви можете зашифрувати хеш-код з використанням секретного ключа, недоступного зломщикові. Цей підхід називається алгоритмом генерування зашифрованого хеш-коду. **Microsoft .NET Framework** підтримує два таких алгоритми (таблиця 2); відповідні їм класи походять від класу **KeyedHashAlgorithm** з підпростору імен **System.Security.Cryptography**. Їх головна відмінність від звичайних класів алгоритмів генерування хешу в тому, що вони включають властивість Key, що служить для зберігання секретного ключа (послідовності байт), який знадобиться при генерації хешу.

Таблиця 2

Класи алгоритмів генерування зашифрованого хеш-коду

Алгоритм	Клас реалізації за умовчанням	Розмір хешу (у бітах)	Розмір ключа
HMAC-SHA1	MAC-3DES-CBC	160	64 (рекомендується)
HHACSHA1	HACTripleDES	64	8, 16 або 24

Наступний фрагмент генерує зашифровані хеш-коди для одних і тих же даних, використовуючи два різних ключа. Зверніть увагу: підсумкові хеш-коди будуть різними.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
' Створення нового файлу.
    Dim fs As New FileStream(«testfile,bin», FileMode.Create)

' Запис даних.
    Dim w As New StreamWriter(fs)
    w.WriteLine("This is the first line.")
    w.WriteLine("This is the second line.")
    w.Flush()

' Обчислення зашифрованого хэш-коду для файлу.
    Dim HMACSHA As New HMACSHA1()
    fs.Position = 0
    Dim HASHA() As Byte = HMACSHA.ComputeHash(fs)

' Відображення хешу.
    TextSource1 = BitConverter.ToString(HASHA)

'Обчислення зашифрованого хешу по тому ж алгоритму
'для тих же даних, але з використанням іншого ключа,
    fs.Position = 0
    HMACSHA = New HMACSHA1()
    Dim HASHB() As Byte = HMACSHA.ComputeHash(fs)

' Відображення хешу.
    TextSource2= BitConverter.ToString(HASHB)

    fs.Close()

' Порівняння хеш-кодів.
    If CompareByteArray(HASHA, HASHB) Then
        MessageText = "Хеш-коди ідентичні!"
    Else
        MessageText = "Невідповідність хеш-кодів!"
    End If

End Sub
```

СТВОРЕННЯ ЦИФРОВОГО ПІДПИСУ ДЛЯ XML-ДОКУМЕНТУ

Для шифрування XML-вмісту файлу або масиву байт можна використовувати способи хешування і підписання даних, розглянуті в п. 3, так само, як і при генерації хешу для даних будь-якого іншого типу. Проте з цим підходом пов'язані перераховані далі обмеження.

Немає простого способу створити хеш-код і потім перетворити отриманий документ (початковий код XML і хеш-код) в коректний XML-документ.

Незначні відмінності XML-документів (такі як додаткові порожні місця) приводитимуть до зміни хеш-коду, не впливаючи на синтаксичний аналіз

XML-даних. Це може стати особливо серйозною проблемою, якщо створений в одній системі хеш-код вам потрібно перевіряти в додатку, написаному на іншій мові програмування, або за допомогою інших інструментів XML

Немає простого способу підписання тільки частини XML-документа (або підписання різних його частин підписами різних користувачів).

Вирішення цих проблем описані в рекомендації **XML Signatures** консорціуму **W3C** за адресою *http://www.w3c.org/xmldsig-core*. Рекомендація **XML Signatures** визначає стандартний спосіб підписання XML-документів і канонічну репрезентацію **XML**, яка потрібна для того, щоб ідентичні документи завжди генерували однакові підписи. Проте стандарт **XML Signatures** визначає ті ж алгоритми підписання, що використовуються для даних будь-яких інших типів.

У **.NET Framework** стандарт **XML Signatures** реалізований у вигляді типів з простору імен **System.Security.Cryptography.Xml**. Для доступу до цих типів ви повинні додати посилання на збірку **System.Security.dll**.

Існують три типи підписів XML

1) відособлений (detached) зберігається як окремий XML-фрагмент, що посилається на підписаний документ за допомогою **URI** (Uniform Resource Identifier);

2) обгорнутий (enveloping) грає роль оболонки для XML-документа;

3) обернутий (enveloped) вставляється в початковий XML-документ, що містить підписувану інформацію.

.NET дозволяє створювати підписи всіх цих типів. Нижче приведений код Windows-додатку, який створює і верифікує обернені підписи. Ось код початкового XML-документа:

```
<Orders>
<Order>
     <Name>Toaster Oven</Name>
     <Price>400.99</Price>
</Order>
</Orders>
```

Підписаний XML-документ складається із початкових XML-даних і підпису, що включає інформацію про алгоритм, який був використаний для її створення, а також відкритий ключ, потрібний для її верифікації.

```
Private Sub Signing()
' Завантаження XML-даних, які ви хочете підписати.
    Dim Doc As New XmlDocument
    Doc.Load("doc.xml")

' Створення підпису XML.
    Dim SignedXml As New SignedXml(Doc)

' Підписання даних всього документа. Ви могли б скласти цей вираз XPath
' так, щоб воно указувало на один елемент.
    Dim Reference As New Reference()
```

```
    Reference.Uri = "#xpointer(/)~"
    SignedXml.AddReference(Reference)

' Використовується перетворення, потрібне для обернутих підписів.
    Reference.AddTransform(New XmlDsigEnvelopedSignatureTransform())

' Для створення підпису використовується алгоритм RSA.
    Dim Rsa As New RSACryptoServiceProvider()

' Додавання інформації про ключ в підпис.
    SignedXml.SigningKey = Rsa
    Dim KeyInfo1 As New KeyInfo
    KeyInfo1.AddClause(New RSAKeyValue(Rsa))
    SignedXml.KeyInfo = KeyInfo1

' Обчислення підпису.
    SignedXml.ComputeSignature()

' Отримання XML-репрезентації підпису.
    Dim XmlSignature As XmlElement = SignedXml.GetXml()

' Вставка XML-підпису в документ.
    Dim Node As XmlNode = Doc.ImportNode(XmlSignature, True)
    Dim Root As XmlNode = Doc.DocumentElement
    Root.InsertAfter(Node, Root.FirstChild)

' Збереження XML-документа, що містить обернутий підпис.
    Doc.Save("SignedDoc.xml")

' Відображення повного документа разом з підписом.
    TextSource = Doc.OuterXml
End Sub
```

Верифікація підписаних даних

```
Private Sub Verifying()
' Завантаження підписаного XML-документа.
    Dim Doc As New XmlDocument()
    Doc.Load("SignedDoc.xml")

' Створення об"єкту SignedXml для верифікації підпису.
    Dim SignedXml As New SignedXml(Doc)

' Знаходження першого підпису.
    Dim Node As XmlNode = Doc.GetElementsByTagName("Signature",
       "http://www.w3.org/2000/09/xmldsig#")(0)
    SignedXml.LoadXml(Node)

' Верифікація підпису.
    If SignedXml.CheckSignature() Then
       MessageBox.Show("Signature authenticated.")
    Else
       MessageBox.Show("Invalid signature.")
    End If
End Sub
```

Практичне завдання

1. Обчислити значення хеш-функції для довільного документу MS Word.

   Варіант 1 – хеш-функція MD5
   Варіант 2 – хеш-функція SHA-1
   Варіант 3 – хеш-функція SHA-256
   Варіант 4 – хеш-функція SHA-512

2. Створити захищений від підміни хеш-код для одного документу до і після його модифікації. Перевірити відповідність хеш-кодів.
3. Створити XML-документ, який містить дані про студентів групи (Прізвище, Ім.'я, Вік, Стать…) та підписати його цифровим підписом.
4. Перевірити правдивість отриманих XML-даних.

Контрольні запитання

1. Що таке цифровий підпис?
2. Для чого призначений цифровий підпис?
3. Які умови повинні виконуватися при передачі даних?
4. Яка послідовність формування цифрового підпису?
5. Що таке хеш-функція?
6. Яким умовам повинна відповідати хеш-функція?
7. Які криптоалгоритми найчастіше використовуються для генерування цифрового підпису?
8. Яким чином можна перевірити цілісність прийнятого повідомлення?
9. В яких стандартах визначені алгоритми перевірки достовірності та цілісності повідомлень?

Лабораторне заняття № 8

ЗАХИСТ КОМП'ЮТЕРНИХ МЕРЕЖ

Мета: Ознайомитися із способами та методами захисту комп'ютерних мереж від несанкціонованого втручання та пошкоджень.

Необхідне програмне забезпечення

Брандмауери Outpost Firewall Pro, Kaspersky Antihacker, ZoneAlarm Pro. Мережева утиліта Net Tools. Пакет аналізу недоліку системи безпеки Tiger Tools.

Теоретична частина

ТИПИ АТАК В КОМП'ЮТЕРНИХ МЕРЕЖАХ

Несанкціонований доступ до терміналу може привести до:

- розкрадання інформації, порушення її конфіденційності
- зміни інформації або роботи від чужого імені, частіше саме ця загроза приводить до розкрадання грошових коштів.

Загрози безпеці, можливі наслідки і заходи протидії на АРМ віддалених клієнтів.

Робота від імені іншого клієнта (абонента). Може привести, зокрема, до успішного здійснення розкрадання грошових коштів.

Заходи протидії: Застосування надійних засобів аутентифікації віддалених клієнтів.

Загрози безпеці інформації, можливі наслідки і заходи протидії при роботі з комп'ютерною мережею.

1. Перехоплення інформації, що передається по комп'ютерній мережі може привести до порушення конфіденційності фінансової (господарською) інформації підприємства, клієнта або банку.

Заходи протидії: Застосування засобів криптографічного захисту інформації.

2. Модифікація інформації, що передається по комп'ютерній мережі (підміна, помилкові повідомлення і т.д.).

Заходи протидії: Застосування засобів ЕЦП (електронного цифрового підпису).

3. Робота від чужого імені (наприклад, помилковий видалений клієнт).

Заходи протидії: Надійна аутентифікація всіх учасників інформаційної взаємодії по каналах зв'язку.

Для забезпечення адекватного рівня безпеки в АС в цілях протидії розкраданню грошових коштів та інших шкідливих дій, необхідне комплексне застосування засобів захисту від всіх перерахованих погроз у всіх основних елементах.

ПРОГРАМИ-СКАНЕРИ

Сканер – це програма, призначена для автоматизації процесу пошуку слабкостів в захисті комп'ютерів, підключених до мережі у відповідності протоколом TCP/IP. Найбільш довершені сканери звертаються до портів TCP/IP видаленого комп'ютера і в деталях протоколюють відгук, який вони отримують від цього комп'ютера. Запустивши сканер на своєму комп'ютері, користувач, скажімо, з Боярки, навіть не виходячи з будинку, може знайти дірки в захисних механізмах сервера, розташованого, наприклад, в Лос-Анджелесі.

Зазвичай сканери створюються і використовуються фахівцями в області мережевої безпеки. Як правило, вони розповсюджуються через мережу Internet, щоб з їх допомогою системні адміністратори могли перевіряти комп'ютерні мережі на предмет наявності в них вад. Тому володіння сканерами, так само як і їх використання на практиці, цілком законно.

Часто до сканерів помилково відносять утиліти типу host, Risers, finger, Traceroute, Showmount. Зв'язано це з тим, що, як і сканери, дані утиліти дозволяють збирати корисну статистичну інформацію про мережеві служби на видаленому комп'ютері. Цю інформацію можна потім проаналізувати на предмет виявлення помилок в їх конфігурації.

Дійсно, мережеві утиліти виконують ряд функцій, які характерні для сканерів. Проте на відміну від останніх використання цих утиліт викликає менше підозр у системних адміністраторів. Виконання більшості мережевих утиліт на видаленій хост-машині практично не робить ніякого впливу на її функціонування. Сканери ж часто поводяться як слон в посудній лавці і залишають сліди, які важко не помітити. Крім того, хороший сканер – явище досить рідкісне, а мережеві утиліти завжди під рукою. До недоліків мережевих утиліт можна віднести те, що доводиться виконувати вручну дуже велику роботу, щоб добитися того ж результату, який за допомогою сканера виходить автоматично.

Використання мережевих шлюзів (firewalls)

Підключення комп'ютера до глобальної мережі Інтернет, разом із значним розширенням можливостей отримання користувачем найрізноманітнішої інформації, спричиняє за собою також і деякі незручності, які, при певних обставинах, можуть перерости у великі проблеми. Головна їх причина в тому, що, дістаючи доступ до практично безмежних ресурсів величезної кількості комп'ютерів в Інтернеті, хочеш не хочеш доводиться відкривати доступ до ресурсів свого комп'ютера з боку інших мешканців Мережі. На будь-якому комп'ютері, підключеному до Інтернету:

- можуть запускатися сторонні програми (наприклад, при відображенні Web-сторінок, ActiveX, що містять, або Java-аплети), які можуть виконувати, практично безконтрольно, будь-які дії, у тому числі і деструктивні;
- інші комп'ютери в Мережі можуть отримати або спробувати дістати доступ до дисків і файлів локального комп'ютера;
- може розміщуватися інформація (наприклад, файли Cookie), використовуючи яку, можна отримати відомості про користувача, його переваги і відвідини різних мережевих ресурсів;

- можуть розміщуватися «троянські коні», тобто програми, які посилають важливу конфіденційну інформацію (наприклад, паролі доступу в Інтернет або номери кредитних карток) із зараженого комп'ютера на певні мережеві адреси. Широко поширеним варіантом вторгнення є негласна установка різних програм для видаленого управління комп'ютером жертви. Найшкідливіший різновид «троянських коней» представляють звичайні програми, для яких навіть придуманий окремий термін, – spyware. В більшості своїй це безкоштовні програми з вбудованою системою відображення рекламних банерів, і їх «нелегальна» діяльність полягає, в основному, в зборі і відправці на сайт розробника статистики про переваги користувача (найбільш часто відвідуваних Web-вузлах, що викликали інтерес, банерах і т.п.);
- разом із потрібною, в комп'ютер завантажується і непотрібна інформація, наприклад, рекламні банери, що може істотно збільшити час завантаження сторінки, особливо при використанні достатньо повільного модемного з'єднання.

Для захисту інформації на локальних комп'ютерах, що мають доступ в Інтернет, широко застосовуються програми, звані персональними брандмауерами (firewall). Ці програми призначені для збереження інформації що міститься в системі, як від несанкціонованого доступу ззовні, так і від спроб самостійної передачі якій-небудь інформації з комп'ютера в мережу, а також для захисту мережевих портів від зовнішніх атак. Персональні брандмауери фільтрують весь вхідний і вихідний трафік на основі стандартних або/і визначених користувачем настройок, тим самим забезпечуючи більш-менш надійний захист всієї системи.

Одним із найбільш популярних серед подібного класу програмних засобів є персональний брандмауер **Outpost Firewall.** Поміж стандартних функціональних можливостей він дозволяє прискорити завантаження Web-сторінок шляхом блокування активного їх вмісту (ACTIVEX, Java-аплети, програми на мовах Visual Basic і Java Script) і, найголовніше, рекламних банерів. Так, наприклад, Outpost може видаляти банери не тільки відповідні певному HTML-коду, але і за розміром картинки, а також контролювати файли, що поступають по протоколу SMTP (електронна пошта) чи NNTP (конференції новин). Останнім часом дуже актуальною стала проблема обмеження доступу до певної категорії сайтів. Так, мало кого обрадує, якщо малолітні чада випадково забредуть на, м'яко кажучи, сайти для дорослих. Outpost Firewall реалізує заборону на відвідування тих або інших Web-ресурсів з урахуванням списку заборонених словосполучень або доменних імен, які містяться в системі і створюються користувачем вручну. Для користувачів-початківців дуже важливим може стати можливість вибору української мови інтерфейсу програми, а також те, що вона готова до роботи відразу після установки, не вимагаючи кропіткої роботи за попереднім визначенням правил, що досягається включенням за умовчанням режиму навчання програми.

Практичне завдання

1. За допомогою мережевої утиліти зібрати всю інформацію про мережу.
2. Встановити один із брандмауерів і налаштувати його.
3. За допомогою пакету Tiger Tools або PCSecurity зімітувати декілька можливих атак на віддалений комп'ютер-хост. Занотувати дії або їх відсутність, які будуть проведені брандмауером.

Контрольні запитання

1. Які існують типи атак та способи несанкціонованого доступу до комп'ютерних мереж?
2. Що таке програми-сканери?
3. Що таке програми-аналізатори?
4. Який принцип роботи програм-сканерів?
5. Яку особливість побудови комп'ютерних мереж використовують програми-аналізатори?
6. Які способи захисту від програм-аналізаторів?
7. Для чого призначені програми-брандмауери?
8. Що собою являє технологія NAT?
9. На які типи діляться брандмауери?
10. З яких основних модулів складаються брандмауери?

Лабораторне заняття № 9-10

УПРАВЛІННЯ КОМП'ЮТЕРНОЮ БЕЗПЕКОЮ

Мета: Ознайомитися із методикою аудиту інформаційної системи установи на відповідність стандарту ISO 17799 за допомогою програмного пакету Digital Security Office 2006.

Необхідне програмне забезпечення

Digital Security Office 2006 (система аналізу і управління інформаційними ризиками **ГРИФ** й система розробки і управління політикою безпеки інформаційної системи **КОНДОР**)

Теоретична частина

Ключовим міжнародним стандартом з безпеки інформації є розроблений **Міжнародною організацією стандартів** (International Standards Organization, ISO) **ISO/IEC 17799:2000** – зведення правил і норм управління безпекою в галузі інформаційних технологій.

Деякі уряди вже використовують цей стандарт у внутрішніх мережах і вимагають від постачальників його дотримання. Таке використання може стати моделлю і для законодавчих органів. Стандарт **ISO/IEC 17799:2000** регламентує такі аспекти:

- планування послідовності дій;
- контроль за доступом до системи;
- побудова та обслуговування систем;
- відповідність вимогам;
- захист особистої інформації;
- захист інформації, що належить організації;
- управління комп'ютерним забезпеченням та мережами;
- класифікація ІТ-активів та контроль за ними;
- політика захисту даних.

Digital Security Office 2006 – завершене рішення для комплексного управління інформаційною безпекою компанії.

Digital Security Office 2006 включає систему аналізу і управління інформаційними ризиками **ГРИФ** і систему розробки і управління політикою безпеки інформаційної системи **КОНДОР**.

КОНДОР – система розробки і управління політикою безпеки ІС компанії на основі стандарту ***ISO 17799***. Це сучасний і зручний інструмент для розробки всіх основних положень політики ІБ компанії і управління процесом впровадження цих положень на практиці.

За допомогою програми **КОНДОР** проводиться аудит ІС компанії на відповідність стандарту ***ISO 17799***. На основі даних, отриманих в результаті

проведення аудиту, розробляється політика безпеки компанії і система управління інформаційною безпекою.

Для проведення аналізу ІС компанії на відповідність стандарту інформаційній безпеці ***ISO 17799*** необхідно перевірити, чи виконуються в компанії вимоги стандарту. Залежно від тимчасового періоду ступінь виконання вимог стандарту міняється, тому необхідно проводити аудит періодично через визначені керівництвом компанії проміжки часу.

ГРИФ – інструмент для аналізу захищеності ресурсів інформаційної системи компанії і ефективного управління ризиками.

Аналіз ризиків ІБ здійснюється за допомогою побудови моделі ІС компанії. Розглядаючи засоби захисту ресурсів з цінною інформацією, взаємозв'язок ресурсів між собою, вплив прав доступу груп користувачів, організаційні заходи, модель досліджує захищеність кожного виду інформації.

В результаті роботи алгоритму програма представляє наступні дані:

1. Інвентаризація.
2. Значення ризику для кожного цінного ресурсу компанії.
3. Перелік всіх можливих «слабких» місць, які стали причиною набутого значення ризику.
4. Значення ризику для ресурсів після завдання контрзаходів (залишковий ризик).
5. Ефективність контрзаходів.
6. Рекомендації експертів.

В підсумку системою **ГРИФ** будується докладний звіт про рівень ризику кожного цінного ресурсу ІС компанії, всі причини ризику з докладним аналізом можливих «слабких» місць і оцінкою економічної ефективності всіх можливих контрзаходів.

Практичне завдання

1. Розрахуйте ризик невиконання вимог стандарту ISO 17799 в модельованій ІС. Проаналізуйте отримані результати. Змоделюйте зміни, які слід ввести в ІС для зниження ризику до прийнятного рівня. Проаналізуйте зміни.

Порядок виконання завдання

1. Запустіть систему розробки і управління політикою безпеки інформаційної системи DS Office 2006: Кондор. (встановлено на віртуальній ОС Windows 2000 Server).
2. Створення проекту:
   2.1. Введіть назву нового проекту «Університет».
3. Властивості проекту:
   3.1. Введіть назву об'єкту, відповідального за виконання роботи користувача і його посаду.
   3.2. Змініть вагові коефіцієнти тих вимог стандарту ISO 17799, які, на Ваш погляд, специфічні для модельованої ІС.

4. Моделювання ІС:
   4.1. Відповідайте на питання розділів стандарту ISO 17799. Вкажіть питання, які непридатні до модельованої ІС (тобто питання, що відносяться до бізнес-процесів, яких не існує в установі).
   4.2. Відповідайте на питання розділів стандарту ISO 17799. Вкажіть питання, які непридатні до модельованої ІС (тобто питання, що відносяться до бізнес-процесів, яких не існує в установі).
5. Звіт:
   5.1. Створіть звіт.
   5.2. Проаналізуйте дані звіту.
6. Управління ризиками:
   6.1. Задайте контрзаходи до вимог стандарту ISO 17799. Для цього внесіть зміни в ІС, які спричинять за собою виконання вимог стандарту ISO 17799 і зменшення ризику невиконання вимог. Введіть вартість впровадження контрзаходу і можливе зниження витрат на ИБ.
7. Звіт:
   7.1. Створіть повторний звіт.
   7.2. Проаналізуйте, чи змінився ризик при завданні контрзаходів.
8. Управління періодами:
   8.1. Створіть новий період аудиту.
9. Повторне моделювання ІС:
   9.1. Змоделюйте ІС. При моделюванні слід врахувати всі зміни, які відбулися в системі з моменту останнього проведення аудиту.
10. Звіт:
   10.1. Побудуйте звіт за проектом.
   10.2. Проаналізуйте результати.

2. Долучіть згенерований звіт до електронної версії звіту про виконання лабораторної роботи і відправте його електронною поштою на скриньку викладача із темою листа «ЗВІТ_ЛАБ_9-10 – <Ваше прізвище>».

Контрольні запитання

1. Дайте означення поняття стандарту інформаційної безпеки?
2. Які міжнародні стандарти акредитовані для використання в Україні?
3. В чому полягає аудит інформаційної системи?
4. Які аспекти інформаційної безпеки описуються стандартом ISO 17799?
5. Які класи функцій системи інформаційної безпеки виділені у стандарті ISO 15408?
6. Яке призначення має програма «КОНДОР» зі складу пакету DS Office 2006?
7. Що дозволяє зробити інструмент «ГРИФ» зі складу пакету DS Office 2006?
8. Які ще існують програмні засоби подібні до DS Office 2006?
9. На основі якого стандарту був прийнятий ISO 17799 (його аналог)?
10. Що дає прийняття та затвердження політики безпеки для установи?

ЛІТЕРАТУРА

1. Аграновский А.В., Хади Р.А. Практическая криптография. – М.: СОЛОН-Пресс, 2002. – 256 с.
2. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основі криптографии: Учебное пособие, 2-е изд., испр. и доп. – М.: Гелиос АРВ, 2002. – 480 с., ил.
3. Альманах программиста, том 4. Безопасность в Microsoft .NET / Сост. Ю.Е. Купцевич. – М.: Издательско-торговый дом «Русская Редакция», 2004. – 304 с.
4. Анин Б.Ю. Защита компьютерной информации. – СПб.: БХВ-Петербург, 2000.
5. Кландер Ларс. Hacker proof: Полное руководство по безопасности компьютера. – Минск: Попурри, 2002.
6. Макдональд Метью. Рецепты программирования. Мастер-класс / Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2004. – 704 с.
7. Медведовский Илья. Программные средства проверки и создания политики безопасности, соответствующей ISO 17799. – URL: www.dsec.ru
8. Мельников Ю.Н. Учебное пособие по курсу «Методы и средства защиты информации» / Под. ред. Хорева П.Б. – М.: МЭЕ, 2002.
9. Методические указания по работе с программным комплексом Digital Security Office 2006. – Спб.: Digital Security Company, 2006.
10. Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни «Основи захисту та кодування інформації» студентами спеціальностей 7.080200, 8.080200 «Прикладна математика». Частина І / П.В.Ольшанський. – Рівне: УДУВГП, 2004. – 52 с.
11. Методичні вказівки для виконання лабораторних та самостійних робіт з дисципліни «Основи захисту та кодування інформації» студентами спеціальностей 7.080200, 8.080200 «Прикладна математика». Частина ІІ / П.В.Ольшанський. – Рівне: УДУВГП, 2004. – 60 с.
12. Серикбаев М.Б. Обзор выявления вторжений / Информационная безопасность компьютерных систем, обнаружение вторжений (Intrusion Detection). – СПб.: СПбГТУ ФТК ИБКС (СЦЗИ), 2001.
13. Щербаков Л.Ю., Домашев А.В. Прикладная криптография. Использование и синтез криптографических интерфейсов. – М: Издательско-торговый дом «Русская Редакция», 2003. – 416 с.

ДОДАТКИ

Додаток 1

ЗРАЗОК ОФОРМЛЕННЯ ЗВІТІВ

ЛАБОРАТОРНА РОБОТА № *X*

Тема: <Тема лабораторної роботи>

Мета: <Мета лабораторної роботи>

1. Опис та результати виконання практичного завдання №1
2. Опис та результати виконання практичного завдання №2
3. Опис та результати виконання практичного завдання №3 ……..…
N. Опис та результати виконання практичного завдання № *N*

Висновки: на цьому занятті ми ознайомились (*навчились, з'ясували*)…

Таблиця Віженера для українського алфавіту

А	**Б**	**В**	**Г**	**Д**	**Е**	**Є**	**Ж**	**З**	**И**	**І**	**Ї**	**Й**	**К**	**Л**	**М**	**Н**	**О**	**П**	**Р**	**С**	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ь**	**Ю**	**Я**
Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А
В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б
Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В
Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г
Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д
Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е
Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є
З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж
И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З
І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И
Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І
Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї
К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К
М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М
О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С
У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ь	Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ю	Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь
Я	А	Б	В	Г	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю

Таблиця Бофорта для українського алфавіту

	А	**Б**	**В**	**Г**	**Д**	**Е**	**Є**	**Ж**	**З**	**И**	**І**	**Ї**	**Й**	**К**	**Л**	**М**	**Н**	**О**	**П**	**Р**	**С**	**Т**	**У**	**Ф**	**Х**	**Ц**	**Ч**	**Ш**	**Щ**	**Ь**	**Ю**	**Я**
А	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б
Б	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В
В	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г
Г	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д
Д	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е
Е	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є
Є	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж
Ж	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З
З	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И
И	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І
І	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї
Ї	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й
Й	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К
К	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л
Л	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М
М	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н
Н	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О
О	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П
П	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р
Р	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С
С	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т
Т	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У
У	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф
Ф	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ц	Ю
Х	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ч	Ю
Ц	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ш	Ю
Ч	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Щ	Ю
Ш	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ь	Ю
Щ	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю	Ю
Ь	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я	Ю
Ю	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А	Я
Я	Я	Ю	Ь	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р	П	О	Н	М	Л	К	Й	Ї	І	И	З	Ж	Є	Е	Д	Г	В	Б	А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КАМ'ЯНЕЦЬ-ПОДІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

НАВЧАЛЬНЕ ВИДАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт з дисципліни

«Захист комп'ютерної інформації»

студентам спеціальності 6.080200 «Прикладна математика»

АВТОР-УКЛАДАЧ:

СЛОБОДЯНЮК Олександр Васильович

асистент кафедри інформатики та методики викладання інформатики Кам'янець-Подільського національного університету

Підписано до друку 19.03.2008 р. Формат 60 х 84/16.
Гарнітура Times. Обл. вид. арк. 2,94. Умовн. друк. арк. 3,62.
Зам. № 283. Наклад 100.

Редакційно-видавничий відділ
Кам'янець-Подільського національного університету.
32300, м. Кам'янець-Подільський, вул. Огієнка, 61.
Свідоцтво серії ДК № 117 від 11.07.2000 р.