МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА МОДЕЛЮВАННЯ

Кіндрат П.В.

З А Х И С Т І Н Ф О Р М А Ц І Ї

методичні рекомендації до виконання лабораторних робіт

Рівне 2025

Затверджено на засіданні кафедри інформаційних технологій та моделювання
Протокол від «29» квітня 2025 року № 5

Схвалено навчально-методичною комісією факультету математики та інформатики
Протокол від «30» квітня 2025 року № 4

Рецензент:
Бойчура М.В. к.т.н., доцент.

Кіндрат П.В. Захист інформації: методичні рекомендації до виконання лабораторних робіт [Електронний ресурс] Рівненський державний гуманітарний університет. Рівне. 2025. 28 с.

Методичні рекомендації до виконання лабораторних робіт з курсу «Захист інформації» для студентів спеціальностей 121 «Інженерія програмного забезпечення» та 122 «Комп'ютерні науки» орієнтований на формування практичних навичок щодо застосування методів захисту інформації в практичній площині.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1
«ПРОТИДІЯ ШКІДЛИВОМУ ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННЮ. БОРОТЬБА З MALWARE»

Теоретичні відомості:

Тенденції сучасних комп'ютерних вірусів і боротьба з ними

Можна виокремити характерні риси, які за останні роки найбільш чітко проявилися в сучасних вірусах:

- найбільше поширення отримали мережеві хробаки;
- віруси активно використовують уразливості в різних операційних системах і програмному забезпеченні;
- для швидкого поширення вірусів використовуються спам-технології;
- один вірус поєднує в собі безліч технологій: поліморфних, стелс, бекдор;
- замість пересилання свого тіла по електронній пошті часто відправляється посилання на веб-сайт або на заражений раніше комп'ютер;
- збільшується кількість вірусів для нових платформ: стільникових телефонів, смартфонів і комунікаторів, при цьому активно використовуються бездротові середовища передачі даних (Bluetooth, Wi-Fi).

Непрофесіоналові важко виявити присутність вірусів на комп'ютері, оскільки вони вміло маскуються серед звичайних файлів. Саме тому далі розглянемо ознаки зараження комп'ютера, а також способи відновлення даних після вірусної атаки і заходи щодо запобігання їх ураження шкідливими програмами.

Можна відзначити ряд ознак, що свідчать про зараження комп'ютера шкідливими програмами:

- виведення на екран непередбачених повідомлень або зображень;
- подача непередбачених звукових сигналів;
- зміна дати і часу модифікації файлів
- несподіване відкриття і закриття лотка DVD-ROM-пристрою;
- довільний, без вашої участі, запуск на комп'ютері будь-яких програм;
- зникнення файлів і каталогів або перекручування їхнього вмісту;
- часті зависання і збої у роботі комп'ютера;
- повільна робота комп'ютера;
- неможливість завантаження ОС;
- істотне зменшення розміру вільної оперативної пам'яті;
- припинення роботи або неправильна робота програм, які раніше функціонували нормально;
- зміна розмірів файлів;
- несподіване значне збільшення кількості файлів на диску;
- часте звертання до жорсткого диска (часто блимає лампочка на системному блоці);
- Інтернет-браузер «зависає» або веде себе несподіваним чином (наприклад, вікно програми неможливо закрити);
- при наявності на вашому комп'ютері міжмережевого екрану, поява попереджень про спробу будь-якої з програм вашого комп'ютера вийти в Інтернет, хоча ви це не ініціювали та інше.

Але мало виявити той факт, що комп'ютерна система піддалася впливу шкідливого ПЗ, необхідно виявити та нейтралізувати джерело загрози. Для боротьби з вірусами використовується спеціальне програмне забезпечення - ***антивіруси***. Для кращого розуміння, що таке антивірусне програмне забезпечення, далі розглянемо основні методи виявлення антивірусом своїх жертв.

Метод виявлення, заснований на сигнатурах - метод роботи антивірусів і систем виявлення вторгнень, при якому антивірус, переглядаючи файл (або пакет, який передається по мережі), звертається до словника, в якому містяться сигнатури відомих атак або вірусів. Під *сигнатурою* розуміється фрагмент коду, який однозначно ідентифікує вірус. Наприклад, вірус Email- Worm.Win32.Happy містить рядок «Happy New Year 1999!!», який лише з досить низькою ймовірністю може зустрітися в іншій програмі.

Основний принцип, за яким виділяється сигнатура - вона повинна містити унікальні рядки з цього файлу, настільки характерні, щоб гарантувати мінімальну можливість помилкового спрацьовування. Розробка сигнатур здійснюється вручну шляхом ретельного дослідження декількох файлів, заражених одним вірусом. Автоматична генерація сигнатур (особливо в умовах поліморфних вірусів) поки не дає задовільних результатів.

Кожен сучасний антивірус має велику (кількасот тисяч) базу сигнатур, яка регулярно оновлюється. Проблема виявлення заснована на сигнатурах полягає в тому, що новий вірус (сигнатури якого ще немає в базі) може безперешкодно обійти антивірусний захист. При цьому створення та доставка сигнатури користувачам займає від 11 до 97 годин в залежності від виробника у той час як, теоретично, вірус може захопити весь інтернет менше, ніж за 30 секунд.

Метод виявлення підозрілої поведінки програми. Антивірус простежує поведінку всіх працюючих програм і намагається виявити дії, характерні для вірусу (наприклад, запис даних в exe-файл). Однак цей метод часто викликає помилкові спрацьовування (в результаті користувачі перестають звертати увагу на попередження). Різновид цього методу - *емуляція програми*: перед запуском програми антивірус намагається імітувати його поведінку з метою відстеження підозрілих дій. Даний метод найбільш вимогливий до ресурсів.

Метод «білого списку». Запобігання виконанню всіх комп'ютерних кодів крім тих, які були раніше позначені системним адміністратором як безпечні.

Евристичне сканування - метод, заснований на сигнатурах і евристиці, покликаний поліпшити здатність сканерів застосовувати сигнатури і розпізнавати модифіковані версії вірусів в тих випадках, коли сигнатура збігається з тілом невідомої програми не на 100 %, але в підозрілій програмі присутня більшість загальних ознак вірусу. Однак, дана технологія, застосовується в сучасних програмах дуже обережно, так як може підвищити кількість помилкових спрацювань.

Фахівці антивірусних технологій **виділяють п'ять типів антивірусного ПЗ** за функціями, які вони виконують: сканери, монітори, ревізори змін, імунізатори і поведінкові блокатори.

Сканер. Принцип роботи антивірусного сканера полягає в тому, що він переглядає файли, оперативну пам'ять і завантажувальні сектори дисків на предмет наявності вірусних масок, тобто унікального програмного коду вірусу. Вірусні маски (опис) відомих вірусів містяться в антивірусній базі даних сканера, і якщо він зустрічає програмний код, який збігається з одним з цих описів, то він видає повідомлення про виявлення відповідного вірусу.

Програми-детектори забезпечують пошук і виявлення віруси в оперативній пам'яті, на зовнішніх носіях, і при виявленні видають відповідне повідомлення. Розрізняють детектори універсальні і спеціалізовані. *Універсальні* детектори в своїй роботі використовують перевірку незмінності файлів шляхом підрахунку і порівняння з еталоном контрольної суми. Недолік універсальних детекторів пов'язаний з неможливістю визначення причин викривлення файлів. *Спеціалізовані* детектори здійснюють пошук відомих вірусів за їх *сигнатурою* (повторюваної ділянки коду). Недолік таких детекторів полягає в тому, що вони нездатні виявляти всі відомі віруси.

Програми-доктори (фаги) не тільки знаходять заражені вірусами файли, але і «лікують» їх, тобто видаляють з файлу тіло програми-вірусу, повертаючи файли в початковий стан. На початку своєї роботи фаги шукають віруси в оперативній пам'яті, знищуючи їх, і тільки потім переходять до «лікування» файлів. Серед фагів також можна виділяють полі-фаги, тобто програми- доктори, призначені для пошуку і знищення великої кількості вірусів.

Ревізори (CRC-сканери) відносяться до найнадійніших засобів захисту від вірусів. Ревізори запам'ятовують початковий стан об'єктів (програм, каталогів і системних областей диска) незараженої системи і періодично або за бажанням користувача порівнюють поточний стан з вихідним. Виявлені зміни виводяться на екран відео-монітора. Як правило, порівняння станів проводиться відразу після завантаження операційної системи. При порівнянні перевіряються довжина файлу, код циклічного контролю (контрольна сума файлу), дата і час модифікації та інші параметри.

Програми-ревізори мають досить розвинуті алгоритми, виявляють стелс- вірусів і можуть навіть відрізнити зміни версії програми, що перевіряється, від змін, внесених вірусом.

Програми-фільтри *(сторожа)* являють собою невеликі резидентні програми, призначені для виявлення підозрілих дій при роботі комп'ютера, характерних для вірусів. Такими діями можуть бути:

- спроби корекції файлів з розширеннями COM і EXE;
- зміна атрибутів файлів;
- прямий запис на диск за абсолютною адресою;
- запис в завантажувальні сектори диска;
- завантаження резидентної програми.

При спробі якої-небудь програми зробити зазначені дії «сторож» посилає користувачеві повідомлення і пропонує заборонити або дозволити відповідну дію. Програми-фільтри досить корисні, оскільки здатні виявити вірус на самій ранній стадії його існування до початку розмноження. Однак вони не «лікують» файли і диски. Для знищення вірусів потрібно застосувати інші програми, наприклад фаги. До недоліків програм-сторожів можна віднести їх «настирливість» (наприклад, вони постійно видають попередження про будь- яку спробу копіювання виконуваного файлу), а також можливі конфлікти з іншим програмним забезпеченням.

Імунізатори (вакцини) - резидентні програми, що запобігають зараженню файлів. Імунізатори застосовують, якщо відсутні програми-доктори, які «лікують» цей вірус. Вакцинація можлива тільки від відомих вірусів. Вакцина модифікує програму або диск таким чином, щоб це не відображалось на їх роботі, а вірус буде сприймати їх вже зараженими і тому не буде намагатися проникнути. В даний час програми-вакцини мають обмежене застосування.

Істотним недоліком таких програм є їх обмежені можливості щодо запобігання зараження від великої кількості різноманітних вірусів.

Таким чином, на сьогоднішній день перелік доступних антивірусних програм досить широкий. Вони розрізняються як за ціною (від достатньо дорогих - комерційних до абсолютно безкоштовних), так і по своїм функціональним можливостям. Найбільш потужні (і, як правило, найбільш дорогі) антивірусні програми являють собою насправді пакети спеціалізованих утиліт (багатофункціональні програмні комплекси), здатних при спільному їх використанні виявляти, лікувати (видаляти) віруси, а також перешкоджати їх проникненню на комп'ютер.

Сучасні антивіруси можуть працювати в двох режимах. В режимі *монітора* антивірус постійно працює, відстежуючи всі звернення до файлів, вклинюючись в цей процес і перевіряючи ці файли на предмет зараження. Таким чином, при першій спробі вірусу активувати антивірус блокує цю спробу і видає попередження. При використанні режиму монітора робота комп'ютера сповільнюється (так як частина обчислювальних ресурсів витрачається на роботу антивірусу, а будь-яке звернення до файлів і деяким іншим об'єктам супроводжується процедурою сканування). Крім того, якщо на комп'ютері присутні заражені файли, які не проявляють активності і звернення до них не відбувається, вони залишаться непоміченими.

В режимі *сканера* антивірус перевіряє всі файли в заданій області (певний каталог, розділ жорсткого диска або всі пристрої зберігання інформації) і видаляє/лікує заражені (або просто сповіщає про них - в залежності від налаштувань сканера). Перевірка всіх даних на

комп’ютері може зайняти значний час (кілька годин). Крім того, вірус може потрапити в систему відразу після сканування.

Для надійного захисту рекомендується застосування обох режимів: постійна робота антивірусу в режимі монітора і регулярна (наприклад, раз в тиждень) перевірка всіх даних за допомогою сканера (зазвичай сканування запускається на ніч).

Однак необхідно відзначити, що 100 % захисту жодне антивірусне ПЗ не забезпечує. Це пов’язано з тим, що будь-який розробник антивірусного ПЗ може оновити свою вірусну базу лише після того, як хтось з постраждалих надішле файл заражений раніше невідомим вірусом. Після цього потрібно ще деякий час для аналізу та створення сигнатури, за допомогою якої можливе знешкодження нового вірусу і час на оновлення антивірусних баз користувачів. У кращому випадку це приблизно 12 годин, за які новий вірус може встигнути поширитися по мережі Інтернет. Але, тим не менш, застосовувати антивірусне ПЗ необхідно, тому що крім нових вірусів система Інтернет наповнена великою кількістю старих вірусів, від яких вже захист є гарантованим.

До найбільш потужних і популярних антивірусних пакетів сьогодення відносяться: антивірус Касперського (AVP), Doctor WEB, NOD32, Norton Antivirus корпорації Symantec, McAfee VirusScan від компанії Network Associates, Panda Antivirus, Avast! Antivirus (останній є безкоштовним для домашнього використання).

При зараженні комп'ютера хробаки зазвичай виробляють наступні дії:

Створюють виконуваний файл з розширенням *.exe з довільним ім'ям або ім'ям дуже схожим на ім'я системних файлів Windows. У деяких черв'яках можуть використовуватися технології притаманні вірусам, в такому випадку черв'яки інфікують вже існуючий файл програми (наприклад WSOCK32.DLL) або замінюють його на свій файл (наприклад I-Worm.MTX записує одну зі своїх процедур в файл WSOCK32.DLL таким чином, що вона перехоплює відсилання даних в Інтернет (процедура send). Внаслідок черв'як в зараженій бібліотеці WSOCK32.DLL отримував управління кожен раз, коли дані відправляються в Інтернет).

Крім цього, разом з додаванням в систему виконуваних файлів, в ряді випадків черв'яки поміщають в систему файли бібліотек, які зазвичай виконують функції Backdoor компонентів (наприклад один з клонів хробака MyDoom - I-Worm.Mydoom.aa створював системному каталозі Windows файл tcp5424.dll , що є Backdoor-компонентом і реєстрував його в системному реєстрі: HKCR \CLSID \ {E6FB5E20-DE35-11CF-9C87-00AA005127ED} \ InProcServer32 {Default} = "% SysDir% \tcp5424.dll").

Шкідлива програма може вносити зміни в системні файли win.ini і system.ini. Наприклад Email-Worm.Win32.Toil при установці в заражаємо системі копіює себе в папку Windows з випадковим ім'ям і записує у файл system.ini наступні значення:

[Boot]

shell = Explorer.exe% ім'я хробака%

що забезпечує йому автозапуск при кожному перезавантаженні Windows (але тільки під Win9x/Me).

Email-Worm.Win32.Atak.h в процесі інсталяції копіює себе з ім'ям dec25.exe в системний каталог Windows і модифікує файл win.ini для свого подальшого запуску - додає повний шлях до файлу dec25.exe в ключ run секції [windows]:

[Windows]

run =% SystemDir% \ dec25.exe)

Слід так само відзначити, що у файлі system.ini крім секції [boot] шкідливі програми можуть використовувати секцію [Drivers].

Шкідливі програми можуть вносити зміни в наступні гілки реєстру:

HKEY_LOCAL_MACHINE \ Software \ Microsoft \ Windows \ CurrentVersion в ключі Run, RunOnce, RunOnceEx, RunServices, RunServicesOnce - для того щоб система запускала створені хробаком файли.

HKEY_CURRENT_USER \ Software \ Microsoft \ Windows \ CurrentVersion в ключ Run.

Наприклад, Email-Worm.Win32.Bagle.ax після запуску копіює себе в системний каталог

Windows, після чого реєструє в реєстрі скопійований файл: HKEY_CURRENT_USER \ Software \Microsoft \ Windows \ CurrentVersion \ Run "Sysformat" = "% System% \ sysformat. exe

HKEY_CLASSES_ROOT \ exefile \ shell \ open \ command

Крім вище перерахованих гілок і ключів реєстру шкідливі програми можуть вносити зміни і в інші гілки і ключі реєстру, наприклад:

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ WOW \boot

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ Drivers32

HKEY_LOCAL_MACHINE \ Software \ Microsoft \ Windows NT \ CurrentVersion \ WinLogon

HKEY_LOCAL_MACHINE \ SYSTEM \ CurrentControlSet \ Services

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Active Setup \ Installed Components

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows NT \ CurrentVersion \ AeDebug.

Усунення наслідків зараження

У зв'язку з тим що, черв'яки практично не використовують технології шифрування і метаморфізму для маскування своїх копій, боротьба з ним вручну дещо спрощується і зводиться до наступного алгоритму дій:

1. Аналіз і виявлення за допомогою Диспетчера завдань Windows підозрілих процесів.
2. Аналіз відкритих портів за допомогою команди Netstat.
3. Вивантаження підозрілих процесів.
4. Аналіз реєстру за допомогою утиліти Regedit.exe в вище перерахованих гілках і ключах.
5. Відновлення та правка ключів реєстру.
6. Пошук інфікованих файлів по імені на основі даних аналізу процесів операційної системи і даних аналізу реєстру.
7. Видалення або заміна інфікованих файлів.
8. Перевантаження системи.
9. Контрольний аналіз процесів, ключів реєстру, відкритих портів.

Якщо підозрілих процесів не виявлено, ключі реєстру не змінилися, значить, процедуру дезінфекції комп'ютера можна вважати успішною, в іншому випадку алгоритм доведеться повторити. Тим не менше, враховуючи той факт, що сучасні хробаки можуть використовувати технології притаманні вірусам, встановлювати backdoor-компоненти, ускладнюючи тим самим, процедуру аналізу та виявлення своїх файлів, для повної впевненості комп'ютер після дезінфекції вручну рекомендується здійснити перевірку антивірусним засобом.

Слід також зазначити, що боротися вручну з шкідливими програмами можна тільки постфактум - після того, як вони вразили комп'ютер, в той же час використання антивірусного програмного забезпечення в переважній більшості випадків не допустить активації шкідливої програми на комп'ютері.

Проблеми захисту від клавіатурних шпигунів (Keylogger)

Клавіатурний шпигун – це програмний або апаратний засіб для непомітного запису інформації під час натискання клавіш користувачем. А отже, у певних ситуаціях клавіатурні шпигуни можуть вважатися шкідливим програмним забезпечення і служити для зламу захищених програм. Основними принципами побудови клавіатурних шпигунів є такі:

а) стандартна клавіатурна пастка – хук перехоплення повідомлення від клавіатури;

б) періодичне опитування стану клавіатури – циклічне опитування стану клавіатури з великою швидкістю;

в) встановлення драйвера-фільтра – підключення до стеку клавіатури;

г) шпигун-руткіт – перехоплення обміну процесу csrss.exe драйвером клавіатури або стеження за викликами APIфункцій;

д) апаратний «жучок» – пристрій для непомітного запису інформації при натисканні клавіш користувачем.

Сучасні кейлогери в процесі своєї роботи використовують бібліотеки або драйвери, опитують клавіатуру, перехоплюють процеси, мають можливість передавання виявленої інформації по мережі Інтернет та використовують руткіт-метод для забезпечення своєї непомітності в системі.

Основні методи виявлення клавіатурних шпигунів:

- пошук по сигнатурах – перевірка сигнатур файлів;
- евристичні алгоритми – перевірка програми по ознаках, що характерні шпигунам;
- моніторинг API-функцій – перехоплення ряду функцій, що зазвичай використовуються клавіатурними шпигунами;
- відстеження використовуваних системою драйверів, процесів і сервісів.

Найпопулярніші антикейлогери не забезпечують повного захисту від клавіатурних шпигунів. Крім того, вони далеко не ідеальні у тому сенсі, що при їх роботі можливі помилкові спрацювання, оскільки такі самі функції використовуються й у багатьох звичайних програмах.

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

Завдання 1

Підготувати коротку письмову доповідь по заданому пакету антивірусного програмного забезпечення (згідно варіанту в таблиці), використовуючи будь-які доступні джерела інформації, описати основні функції, переваги та недоліки:

Номер	Назва антивірусного забезпечення
1	Avast Software, Чехія
2	AVG Technologies, Чехія
3	Avira, Німеччина
4	BitDefender, Румунія
5	Emsisoft, Австрія
6	Eset, Словаччина
7	F-Secure, Фінляндія
8	McAfee, США
9	Panda Security, Іспанія
10	Qihoo 360, Китай
11	Trend Micro, Японія
12	Check Point, Ізраїль
13	Zillya!, Україна
14	Microsoft, США

При виконанні завдань 2 та 3 лабораторної роботи на комп'ютерах у навчальній лабораторії чи вдома ! обов'язково використовуйте віртуальну машину !.

Завдання 2

a) Увійти в систему під обліковим записом адміністратора. Всі дії в наступних підпунктах виконуйте в системі, що працює на віртуальній машині.

b) Інсталювати та ознайомитися з усіма функціями і можливостями антивірусного ПЗ згідно свого варіанту;

c) Навчитися грамотної експлуатації встановленої версії антивірусного ПЗ та провести його тестування (випробування). Для тестування можна використовувати сервіс Vxvault.net - сайт зі списком шкідливого програмного забезпечення (рис. 1), який оновлюється щоденно, майже в реальному часі.;

Рис. 1 Бібліотека шкідливого програмного забезпечення Vxvault

В даному списку все шкідливе програмне забезпечення підсвічується трьома кольорами:

- жовтий - означає, що шкідник, скоріш за все, вже видалений;
- темно-жовтий - означає, що шкідник видалений;
- зелений - означає, що шкідник доступний для скачування (Link).

Окрім цього сайт відображає хеш файлу, IP-адресу та вказує базові інструменти для вивчення і дослідження *malware.*

Завантаживши, за посиланням ***Link*** (рис. 2), шкідливе програмне забезпечення, його можна перевірити та отримати більш детальну інформацію використовуючи онлайн інструменти, наприклад *VirusTotal (https://www.virustotal.com/gui/home/upload)* (рис. 3).

Рис. 2 Сторінка бібліотеки Vxvault з конкретним malware

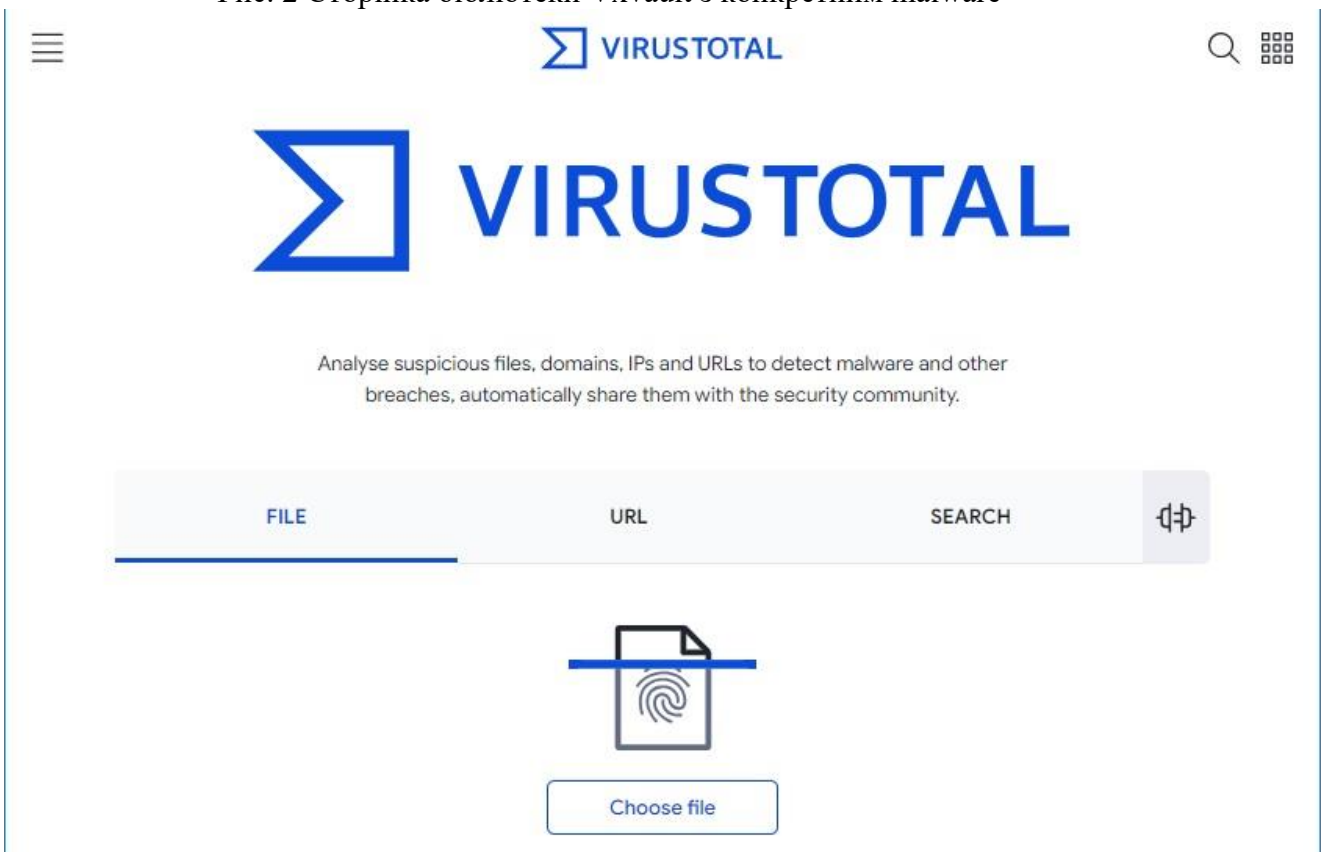

Рис. 3 Сторінка сервісу онлайн аналізу malware за допомогою сервісу VirusTotal

d) Описати виявлене шкідливе програмне забезпечення.

Завдіння 3

a) Завантажте і встановіть програму – клавіатурний шпигун (Actual Spy, NeoSpy, Real Spy Monitor, SpyGo).

b) Запустіть програму і переведіть її у прхований режим роботи.

c) Здійснити короткий серфінг по мережі інтернет, запустити налаштуванн властивносей операційної системи.

d) Відкрити запущений Keylogger, перевірити які дії він зафіксував.

e) Результати запишіть у звіт.

Завдання 4

Встановість антивірусне програмне забезпечення та проскануйте систему.

Перевірте чи виявить обране антивірусне програмне забезпечення завантажені в завданнях 2 і 3 елементи.

Завдання 5

1. Запустіть Диспетчер завдань Windows і проаналізуйте список запущених процесів, навантаження, яку вони надають на ЦП. Запишіть в звіт лабораторної роботи назву підозрілих процесів. Поясніть, на підставі чого були зроблені такі висновки.
2. Використовуючи Диспетчер завдань Windows вивантажити підозрілі процеси.
3. Запустіть утиліту роботи з реєстром Windows. Проаналізуйте вміст перерахованих в теоретичній частині гілок і ключів реєстру.
4. При необхідності внесіть зміни в параметри ключів з метою видалення підозрілих записів, створені шкідливою програмою. Запишіть в протокол лабораторної роботи всі внесені зміни.
5. Проаналізуйте зміст файлу win.ini на предмет підозрілих записів. При необхідності внесіть корективи в файл win.ini. Запишіть в протокол лабораторної роботи внесені коректування.
6. Проаналізуйте файл system.ini на предмет підозрілих записів. При необхідності внесіть корективи в файл system.ini. Запишіть в протокол лабораторної роботи внесені коректування.
7. Використовуючи стандартні засоби пошуку, знайдіть файли, які породжували підозрілі процеси і видаліть їх. Запишіть в протокол лабораторної роботи імена цих файлів.
8. Перезавантажте комп'ютер, виконайте пункти 1-3, щоб переконатися, що видалення шкідливої програми пройшло успішно, в оперативній пам'яті немає підозрілих процесів, ключі реєстру не змінилися, на комп'ютері відсутні підозріло відкриті порти.

Оформити звіт.

ЛАБОРАТОРНА РОБОТА №2
«ПРОБЛЕМИ БЕЗПЕКИ МЕРЕЖЕВИХ-ЗАСТОСУВАНЬ»

Теоретичні відомості:

Міжмережеве екранування

Основним загальновизнаним засобом захисту інформації при підключенні ІТС до мережі Інтернет є міжмережевий екран (*Брандмауер - Firewall*).

Міжмережевий екран (МЕ) виконує функції розмежування інформаційних потоків на кордоні ІТС, що захищається. Це дозволяє:

- підвищити безпеку об'єктів внутрішнього середовища за рахунок ігнорування неавторизованих запитів із зовнішнього середовища;
- контролювати інформаційні потоки в зовнішнє середовище;
- забезпечити реєстрацію процесів інформаційного обміну.

Таким чином міжмережевий екран встановлюється між внутрішньою мережею та мережею Інтернет і виконує роль мережевого фільтра

Контроль інформаційних потоків проводиться за допомогою фільтрації інформації, тобто аналізу її за сукупністю критеріїв (відповідно до політики інформаційної безпеки організації) і прийняття рішення щодо пропуску допустимого трафіку від користувачів мережі до служб мережі Інтернет і назад; та відповідно обмежити трафік з боку мережі Інтернет до внутрішньої мережі, яка потребує захисту, лише необхідними службами, наприклад: smtp, dns, ntp.

Залежно від принципів функціонування та застосування різних технологій мережевого захисту, виділяють кілька класів міжмережевих екранів, а саме:

- пакетний фільтр (Packet-filtering);
- шлюз додатків (Application-gateway);
- брандмауер рівня з'єднання (Circuit-level);
- брандмауер перевірки стану (Stateful Inspection).

Найпростішим класом залишаються ***пакетні фільтри*** в яких фільтрація пакетів зазвичай здійснюється за наступними критеріями:

- IP-адреса джерела;
- IP-адреса одержувача;
- порт джерела;
- порт одержувача;
- специфічні параметри заголовків мережевих пакетів.

Сама ж фільтрація реалізується шляхом порівняння перерахованих параметрів заголовків мережевих пакетів з базою правил фільтрації.

Ще одним досить поширеним класом МЕ є ***шлюз додатків*** - модель більшості персональних брандмауерів. Також дані види брандмауерів іноді називаються *проксі-брандмауерами*, оскільки вони можуть фільтрувати, ґрунтуючись на IP-адресах і певних функціях, які намагається виконати той чи інший додаток. Наприклад, ці брандмауери можуть не пропустити певні програми, такі, як Microsoft NetMeeting, PCAnywhere або FTP. Переглядаючи функції конкретного додатку, брандмауер може навіть дозволити виконувати деякі операції цьому додатку, в той же час блокуючи інші. Один із прикладів цього - FTP-сайт, який дозволяє вам завантажити файли в зазначені директорії без дозволу переглядати або змінювати файли в цих директоріях.

Брандмауери рівня з'єднань пропускають потік трафіку з попередньо схвалених IP-адрес, мереж або постачальників Інтернет-послуг.

Брандмауери перевірки стану замість обмеженої перевірки пакетів на відповідність вимогам порту або додатку вивчають весь пакет цілком і аналізують його вміст. Брандмауери даного класу намагаються визначити тип даних, які передаються і таким чином, якщо розглянуті дані не несуть загрози, брандмауери дозволяють їм пройти.

Технологія NAT

Другою цеглинкою забезпечення захищеності мережі є «заміна мережевої адреси» - Network Address Translation, або NAT (також має назви IP Masquerading, Network Masquerading і Native Address Translation.) - це механізм, який використовується в мережах TCP/IP, що дозволяє перетворювати IP- адреси транзитних пакетів.

Перетворення адрес методом NAT може проводитися майже будь-яким маршрутизуючим пристроєм - маршрутизатором, сервером доступу, міжмережевим екраном, т.д. Найбільш популярним є ***SNAT (source NAT)***, суть механізму якого полягає в заміні адреси джерела (англ. *source*) при проходженні пакета в один бік і зворотної заміні адреси призначення (англ. *destination*) у відповідному пакеті. Поряд з адресами джерело/призначення можуть також замінюватися номера портів джерела і призначення.

Приймаючи пакет від локального комп'ютера, роутер дивиться на IP- адресу призначення. Якщо це локальна адреса, то пакет пересилається іншому локальному комп'ютеру. Якщо ж ні, то пакет пересилається назовні, в мережу Інтернет. Але ж зворотнім адресом у пакеті вказано локальний адрес комп'ютера, який з мережі Інтернет буде недоступний. Саме тому роутер «на льоту» виробляє трансляцію IP-адреси і порту і запам'ятовує цю трансляцію у себе в тимчасовій таблиці. Через деякий час після того, як клієнт і сервер закінчать обмінюватися пакетами, роутер зітре у себе в таблиці запис про *n-ий* порт за строком давності.

Крім *source NAT* (надання користувачам локальної мережі з внутрішніми адресами доступу до мережі Інтернет) часто застосовується також ***destination NAT***, коли звернення ззовні транслюються міжмережевим екраном на комп'ютер користувача в локальній мережі, що має внутрішню адресу і тому недоступний безпосередньо (без NAT) ззовні мережі.

Існує 3 базових концепції трансляції адрес: статична (Static Network Address Translation), динамічна (Dynamic Address Translation) та маскарадна (NAPT, NAT Overload, PAT).

Статичний NAT - трансляція незареєстрованої IP-адреси у зареєстровану IP-адресу на підставі один до одного. Особливо корисно, коли пристрій повинен бути доступним зовні мережі, однак дана трансляція є найбільш слабкою з точки зору безпеки мережі. Оскільки, при цьому, противник безперешкодно «бачить» такий комп'ютер у зовнішній мережі, тому що йому однозначно відповідає певна зовнішня адреса.

Динамічний NAT - трансляція незареєстрованої IP-адреси у зареєстровану адресу від групи зареєстрованих IP-адрес. Тобто внутрішній комп'ютер, виходячи в Інтернет, одержує вільну у цей момент адресу з бази даних. При цьому адреси підмінюються динамічно, і кожне нове *TCP*-з'єднання може бути встановлене з іншою IP-адресою. Це також створює додаткові труднощі противнику, тому що позбавляє його можливості атакувати будь-який внутрішній комп'ютер прицільно.

Перевантажений NAT (NAPT, NAT Overload, PAT) - форма динамічного NAT, який відображає кілька незареєстрованих адрес в єдину зареєстровану IP-адресу, використовуючи різні порти. Відомий також як PAT (Port Address Translation). При перевантаженні кожен комп'ютер у приватній мережі транслюється в ту ж саму адресу, але з різним номером порту. При цьому всі внутрішні комп'ютери можуть працювати з Інтернетом одночасно, а маршрутизатор розрізняє, кому яка відповідь перетрансльовується за службовими даними *TCP*-з'єднання. У зовнішній мережі створюється враження, що до неї звертається тільки один комп'ютер. Така заміна істотно ускладнює життя противнику, тому що повністю приховує внутрішні комп'ютери й перешкоджає «вирахуванню» жертви. Навіть, якщо противник має змогу переглядати трафік, що виходить із внутрішньої мережі, не може визначити, від якого комп'ютера він виходить. Крім того, це виключає можливість ініціативного обігу ззовні до внутрішнього комп'ютера, тому що для маршрутизатора в цьому випадку відсутнє правило прив'язки зовнішньої адреси до внутрішньої. Зокрема виключається можливість сканування ззовні внутрішньої мережі.

Демілітаризована зона

В більшості випадків, організації потрібно мати у себе деякі мережні ресурси, до яких

відкритий доступ з мережі Інтернет. Як правило, це поштовий, dns і web-сервери. Механізм їх роботи допускає, що до них повинен бути дозволений вільний або слабко обмежений обіг з Інтернету. Відповідно ймовірність їх зламу вища, ніж інших комп'ютерів мережі. Із цієї причини розміщати їх усередині зони, що захищається, недоцільно з погляду безпеки, тому що у випадку зламу вони можуть стати воротами для атаки внутрішніх комп'ютерів. Для мінімізації ризику і збереження функціональності такі сервери встановлюють за основним шлюзом мережі, але перед міжмережевим екраном, що забезпечує захист внутрішніх комп'ютерів. Логічну область їх розміщення називають демілітаризованою зоною (рис. 1).

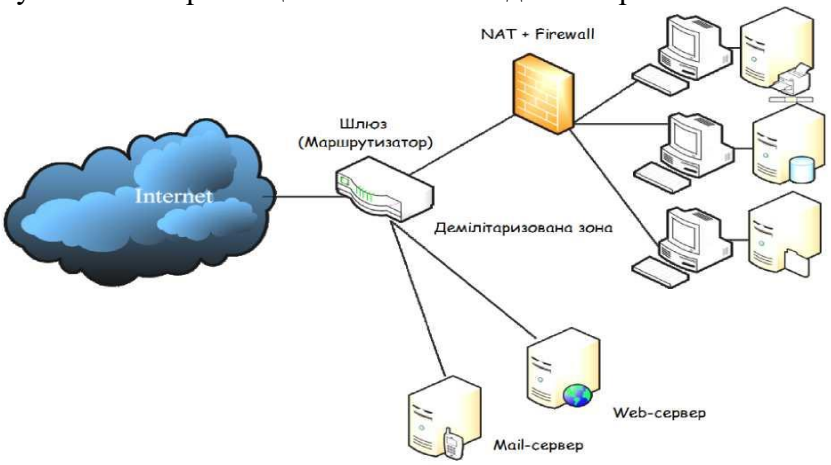

Рис. 1. Структурна схема ІТС з демілітаризованою зоною

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

Завдання 1 Опис політики мережевої безпеки

Підготувати короткі письмові рекомендації щодо забезпечення інформаційної безпеки мережевої інфраструктури підприємства від мережевих загроз.

Параметри мережі:

– розподілена локальна мережа: комп'ютери розміщені в кількох сусідніх будинках;

– наявні файл-сервер та поштовий сервер;

– підприємство здійснює підтримку свого веб-ресурсу на власному апаратному забезпеченні.

Завдання 2 Налаштування Firewall

e) Здійснити налаштування брандмауера засобами операційної системи Windows.

f) Для цього зайти в «Налаштування» -> «Оновлення і безпека» -> Безпека у Windows»

g) Перевірити дозволи програм на роботу через Брандмауер. Визначити приналежність програмних засобів яким у цьому відмовлено .

h) Зайти в меню Windows Defender Firewall:

– Перевірити перелік налаштувань правил для вхідних і вихідних з'єднань.

– Зробити опис правил налаштування з'єднань фаєрволом.

– Звірити перелік програмних за стосунків у яких відсутні дозволи на виконання з'єднань з тими, що не мають дозволу працювати через брандмауер.

– Переглянути логи роботи фаєрволу.

Завдання 3 Налаштування VPN

Здійснити налаштування VPN стандартними засобами Windows 10

Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 3
«НАЙПРОСТІШІ АЛГОРИТМИ ШИФРУВАННЯ ДАНИХ. ШИФР ТРИТЕМІУСА. ШИФР ГАМУВАННЯ.»

Теоретичні відомості

Шифр Тритеміуса – це вдосконалений шифр Цезаря, в якому кожен символ повідомлення зміщується на символ, який відстає від даного на деякий крок. Але крок зміщення робиться змінним, тобто залежним від будь-яких додаткових чинників. Наприклад, можна задати закон зміщення у вигляді лінійної функції позиції літери, що шифрується, або за допомогою використання гасла - текстового рядка, який багаторазово записується під текстом повідомленням.

Таким чином, шифрування і розшифрування для шифру Тритеміуса можна виразити наступними рівняннями:

y=(x+k) mod n x=(y+n-(k mod n)) mod n, де x - символ відкритого тексту, y - символ шифрованого тексту, n - потужність алфавіту.

Крок зміщення k розраховується:

- за лінійним рівнянням k = Ap + B;
- за нелінійним рівнянням k = A 2 + Bp + C ;
- за гаслом.

Тут p - позиція букви в повідомленні. Ключем шифрування виступають відповідно коефіцієнти вказаних рівнянь та гасло.

Метод гамування є розширеною версією ***шифру Віженера*** і полягає в тому, що символи тексту, який шифрується, послідовно складаються з символами деякої спеціальної послідовності, яка називається гамою. Іноді такий метод представляють як накладення гами на вхідний текст, тому він отримав назву «гамування». При цьому символи вихідного тексту і гамми замінюються цифровими еквівалентами, які потім складаються по модулю n, де n - число символів в алфавіті, тобто шифрування і розшифрування для шифру гамування можна виразити наступними рівняннями:

y=(x+g) mod n x=(y+n-(g mod n)) mod n, де x - символ відкритого тексту, y - символ шифрованого тексту, g - символ гами.

Найбільш часто на практиці зустрічається двійкове гамування. При цьому використовується двійковий алфавіт, а складання здійснюється за модулем два: z = x + g (mod 2) = x XOR g.

Операція складання по модулю два в алгебрі логіки називається також "виключне АБО" або поанглійськи XOR. Операція XOR дуже швидко виконується на комп'ютері (на відміну від багатьох інших арифметичних операцій), тому накладення гами навіть на дуже великий відкритий текст виконується практично миттєво.

Цю ж саму операцію використовують і для розшифрування.

При використанні методу гамування ключем є послідовність, з якою проводиться складання – гамма. Якщо гамма коротше, ніж повідомлення, призначене для шифрування, гамма повторюється необхідну кількість разів. Чим довше ключ, тим надійніше шифрування методом гамування.

Розрізняють два різновиди гамування – з кінцевою і нескінченною гамами. При хороших статистичних властивостях гами якість шифрування визначається тільки довжиною періоду гами. При цьому, якщо довжина періоду гами перевищує довжину шифротексту, то такий шифр є абсолютно стійким, тобто його не можна розкрити за допомогою статистичної обробки зашифрованого тексту. При шифруванні за допомогою ЕОМ послідовність гами може формуватися за допомогою генератора псевдовипадкових чисел (ПВЧ).

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

Завдання 1.

Розробіть програмний застосунок криптографічної системи симетричного шифрування, забезпечивши можливість використання в якості ключа:

Варіант 1: 2-вимірного масиву для зберігання коефіцієнтів лінійного рівняння шифрування,

Варіант 2: 3-вимірного масиву для зберігання коефіцієнтів лінійного рівняння шифрування текстового рядка (гасла).

Завдання 2

Адаптуйте інтерфейс криптографічної системи симетричного шифрування для реалізації шифрування методом гамування.

Завдання 3

Сформулюйте систему класів необхідних для реалізації симетричного шифрування методом Тритеміуса, та методом Віжінра передбачивши в них методи валідації ключа, валідації шифрування і розшифрування даних.

Оформити звіт.

ЛАБОРАТОРНА РОБОТА №4
«СИМЕТРИЧНІ КРИПТОСИСТЕМИ. СТАНДАРТ DES»

Теоретичні відомості

Алгоритм симетричного шифрування DES (Data Encryption Standard) - стандарт симетричного шифрування США, розроблений у 1977 році, який згодом набув міжнародного застосування. Зараз DES вважається ненадійним в основному через малу довжину ключа.

З метою забезпечення більш високого рівня криптостійкості був запропонований модифікований метод з послідовним трициклічним шифруванням за алгоритмом DES, який отримав назву 3-DES (TripleDES). Криптостійкість методу виявилась значно кращою (про реалізацію успішних атак на 3DES не відомо), проте швидкість шифрування значно зменшилась у порівнянні з DES (приблизно в 3 рази).

На сьогодні алгоритми DES та 3-DES поступово витісняються новітнім алгоритмом шифрування AES (Advanced Encryption Standard), який забезпечує як високий рівень криптостійкості, так і прийнятну швидкість шифрування.

Нові можливості для розробки криптографічних додатків надає бібліотека класів .NET Framework, яка включає класи криптопровайдерів для реалізації симетричного шифрування за чотирма алгоритмами: DES, TripleDES і AES, а також RC2 (який є попередником AES і залишений для забезпечення сумісності з попередніми версіями додатків). Реалізуються вони за допомогою об’єктів двох класів з простору імен System.Security.Cryptography:

CryptographicServiceProvider - клас, що надає крипто провайдери для кожного з вказаних алгоритмів.

CryptoStream- клас для роботи з криптографічним потоком.

Застосування цих основних об’єктів вимагає використання об’єкту FileStream з простору імен **System.IO**. Крім того, для бітового представлення текстових даних необхідні об’єкти класів UnicodeEncoding (або ASCIIEncoding) з простору імен System.Text.

***Порядок шифрування за їх допомогою такий*:**

1. Створюється потрібний крипто провайдер і задаються його ключ і вектор ініціалізації:

DESCryptoServiceProvider cryptic = new DESCryptoServiceProvider();

cryptic.Key = ASCIIEncoding.ASCn.GetBytes("ABCDEFGH"); cryptic.IV = ASCnEncoding.ASCILGetBytes("ABCDEFGH");

cryptic.Mode = CipherMode.CBC;

2. Відкривається звичайний файловий потік для запису зашифрованих даних:

FileStream stream = new FileStream(@"d:\test.txt",

FileMode.OpenOrCreate,FileAccess.Write)

3. Відкритий файловий потік трансформується в крипто потік для запису:

CryptoStream crStream = new CryptoStream(fs,

cryptic.CreateEncryptor(),CryptoStreamMode.Write);

4. Дані для шифрування перетворюються у бітову послідовність і всі біти (від 0 до data.Length) записуються в крипто потік за допомогою методу Write ():

byte[] data = ASCIIEncoding.ASCII.GetBytes("Hello World!"); crStream.Write(data,0,data.Length);

5. Використані файловий і крипто - потоки закриваються: crStream.Close();

fs.Close();

Порядок розшифрування такий:

1. Створюється потрібний крипто провайдер і задаються його ключ і вектор ініціалізації :

DESCryptoServiceProvider cryptic = new

DESCryptoServiceProvider(); cryptic.Key =
ASCIIEncodmg.ASCII.GetBytes("ABCDEFGH");
cryptic.IV = ASCIIEncoding.ASCII.GetBytes("ABCDEFGH");
cryptic.Mode = CipherMode.CBC;

2. Відкривається звичайний файловий потік для читання зашифрованих даних:
FileStream stream = new FileStream(@"d:\test.txt",
FileMode.Open,FileAccess.Read)

3. Відкритий файловий потік трансформується в криптопотік для читання:
CryptoStream crStream = new CryptoStream(stream, cryptic.CreateDecryptor(),CryptoStreamMode.Read).

4. Дані з крипто потоку зчитуються за допомогою об’єкта StreamReader і присвоюються текстовій змінній:
StreamReader reader = new
StreamReader(crStream); string data = reader.ReadToEnd();

5. Значення текстової змінної виводиться на екран і зчитувач та потік закриваються:
reader.Close(); stream.Close();

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

1. Розробіть інтерфейс криптографічної системи для шифрування за допомогою DES з використанням всіх можливих режимів.
2. Ознайомтесь з описом класів CryptographicServiceProvider і CryptoStream бібліотеки .NET Framework.
3. Реалізуйте шифрування DES, використовуючи класи .NET Framework.
4. Виконайте тестування роботи системи.

Оформити звіт.

ЛАБОРАТОРНА РОБОТА №5
«АСИМЕТРИЧНІ КРИПТОСИСТЕМИ. СТАНДАРТ RSA»

Теоретичні відомості

Шифр RSA отримав назву на честь його розробників Ріверса (Ron Rivers), Шаміра (Adi Shamir) і Адлемана (Leonard Adleman). В RSA системі використовуються наступні факти з теорії чисел:

1. адача перевірки числа на простоту є порівняно простою.
2. Задача розкладання числа n=p*q, де p і q - прості числа, на множники є дуже складною задачею, якщо ми знаємо тільки n, а p і q - великі числа (задача факторизації).

Основні повідомлення між сторонами B і A в протоколі RSA представляються наступною діаграмою:

A↔B: N=PQ, P,Q -прості;
B: f=(P-1)(Q-1); d<f, взаємно просте з f ; cd mod f=1;
B⟶A: d;
A:m; A⟶B: e=m^d mod N
B: y; B⟶A: m'=e^c mod N;

Алгоритм гарантує, що m'=m. Пара чисел (c,N) є секретним, а (d,N) - публічним ключем сторони B.

На платформі .NET алгоритм RSA реалізується за допомогою об'єктів класу **RSACryptoServiceProvider** з простору імен **System.Security.Cryptography**.

Генерація відкритого та закритого ключів здійснюється при створенні нового екземпляра класу. Після створення нового екземпляра класу можна отримати інформацію про ключ одним із двох способів:

1. Метод **ToXMLString** - повертає інформацію про ключ в форматі XML.
2. Метод **ExportParameter**s - повертає структуру RSAParameters, що містить ключові відомості.

Обидва методи приймають як параметр логічне значення, яке показує: false - слід повертати відомості тільки про відкритий ключ; true - слід повертати відомості і про відкритий, і про закритий ключі.

Ініціалізація класу RSACryptoServiceProvider може бути здійснена також двома шляхами:

1. Метод **FromXmlString** - використовує дані ключа з рядка XML.
2. Метод **ImportParameters** - використовує дані структури RSAParameters.

Асиметричні закриті ключі ніколи не повинні зберігатися в роздрукованому вигляді або у вигляді простого тексту на локальному комп'ютері. Якщо необхідно зберігати закритий ключ, слід використовувати для цього контейнер ключа. Контейнер ключа представляє собою екземпляр класу *CspParameters* (з простору імен *System.Security.Cryptography*). В полі **CspParameters.KeyContainerName** задається ім'я контейнера.

Порядок розшифрування за допомогою об'єктів класу RSACryptoServiceProvider такий:

1. Створюється контейнер для збереження ключів:

```
CspParameters cp = new CspParameters(); cp.KeyContainerName = "Key Name";
```

2. Створюється екземпляр криптопровайдера з розміщенням ключів у контейнері:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(cp)
```

3. Публічний ключ експортується для передачі іншій стороні:

```
string pubKey = rsa.ToXmlString(false);
Console.WriteLine("Public Key: \n {0}", pubKey);
```

4. Після отримання байтових даних byte[] EncryptBytes, зашифрованих за допомогою публічного ключа, здійснюється їх розшифрування за допомогою закритого ключа:

```
byte[] DecryptBytes = rsa.Decrypt(EncryptBytes, false); string decryptStr =
Encoding.Unicode.GetString(DecryptBytes);
//string decryptStr =BitConverter.ToString(DecryptBytes);
Console.WriteLine("Decripted string: \n {0}", decryptStr);
```

Порядок шифрування полягає у такому:

1. Створюється екземпляр криптопровайдера :

```
RSACryptoServiceProvider rsa1 = new RSACryptoServiceProvider()
```

2. Імпортується публічний ключ: rsa1.FromXmlString(pubKey);
3. Текст повідомлення перетворюється у байтову послідовність і зашифровується публічним ключем:

```
string dataToEncrypt = "Data to encrypt";
byte[] byteToEncript = Encoding.Unicode.GetBytes(dataToEncrypt);
byte[] EncryptBytes = rsa1.Encrypt(byteToEncript, false);
```

4. Зашифрована байтова послідовність відправляється стороні, яка має для розшифрування відповідний закритий ключ.

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

1. Розробіть інтерфейс криптографічної системи для шифрування з використанням RSA, передбачивши окремий діалог для формування відкритого ключа.
2. Розробіть методи, які б забезпечували:
   - Генерацію пари «відкритий -закритий» ключі.
   - Шифрування з використанням відкритого ключа.
   - Розшифрування з використанням закритого ключа.
3. Перевірте правильність роботи системи на основі використання даних з чисельного прикладу.

Оформити звіт.

ЛАБОРАТОРНА РОБОТА №6
«ПРОГРАМНА РЕАЛІЗАЦІЯ ЦИФРОВОГО ПІДПИСУ»

Теоретичні відомості:

Цифровий підпис

Електронно-цифровий підпис (ЕЦП) - вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача.

ЕЦП накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа.

Схема застосування цифрового підпису (Рис.1):

1. Беремо вихідне повідомлення і створюємо 160-бітовий геш (дайджест повідомлення) за допомогою алгоритму SHA-l.
2. Потім дайджест повідомлення шифрується за допомогою секретного ключа, відомого тільки його власнику, тобто відправнику повідомлення.
3. Будь-який бажаючий може розшифрувати геш, користуючись загальнодоступним відкритим ключем.
4. Результат цього шифрування і називають цифровим підписом.
5. Підписане повідомлення формується об'єднанням вихідного повідомлення, його цифрового підпису та відкритого ключа.

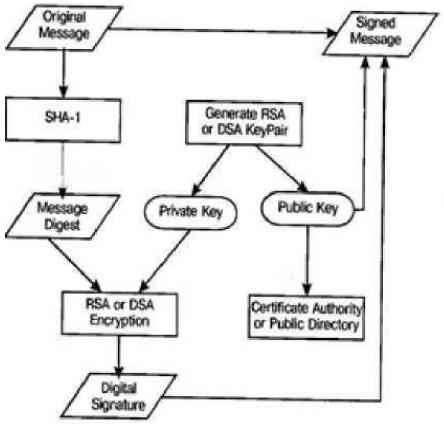

Рис.1. Схема створення цифрового підпису

Схема перевірки цифрового підпису (рис.2):

6. Отримане повідомлення розбивається на три компоненти: на вихідне повідомлення, відкритий ключ і цифровий підпис.
7. Заново обчислюється геш повідомлення.
8. Якщо обчислений геш збігається з розшифрованим гешем, то перевірка підпису пройдена.

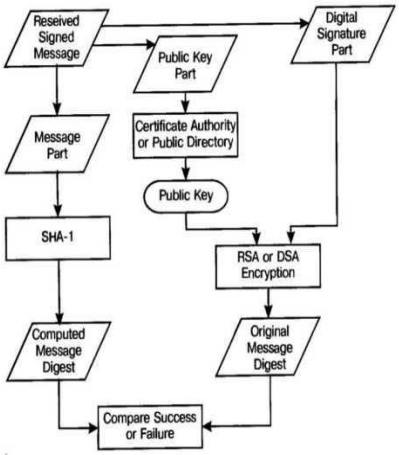

Рис.2. Схема перевірки цифрового підпису

Цифровий підпис на основі шифру RSA

Створити цифровий підпис можна на основі схеми RSA. Так, якщо Аліса має намір використовувати цифровий підпис, вона:

1. Вибирає два великих простих числа P і Q.
2. Обчислює N=PQ і f=(P-1)(Q-1).
3. Вибирає d, взаємно просте з f.
4. Обчислює $c=d^{-1} \bmod f$.
5. Зберігає c в якості секретного ключа (інші числа - P,Q і f більше не потрібні).
6. Публікує N і d в якості відкритого ключа.

Тепер Аліса готова до використання цифрового підпису. Нехай вона хоче підписати повідомлення m= m1, ... , mn. Тоді вона:

7. Спочатку обчислює так називану геш-функцію h =h(m1, ... , mn).

Найбільш важлива особливість цієї функції в тому, що практично неможливо змінити текст m1, . , mn , не змінивши h. Вважається, що

алгоритм її обчислення загальновідомий.

8. Обчислює число s= hc mod N, яке і є цифровим підписом.
9. Додає s до повідомлення m, формуючи підписане повідомлення <m,s> .

Тепер кожний, хто знає відкритий ключ Аліси, може перевірити її справжність. Для цього йому необхідно:

10. Обрахувати геш-функцію h =h(m1, . , mn).
11. Обрахувати $w=s^d \bmod N$..
12. Перевірити виконання рівності w = h.

Неважко переконатись у тому, що у випадку справжнього підпису w = h.

Дійсно, $w=s^d \bmod N= h^{cd} \bmod N= h$.

Приклад. Нехай P=5 і Q=11. Тоді N=5*11=55, f=4*10=40. Візьмемо d=3, т.я. gcd(40,3)=1. За допомогою розширеного алгоритму Евкліда знаходимо $c=3^{-1} \bmod 40 =27$.

Нехай значення геш-функції повідомлення y=13. Тоді Аліса знаходить $s = 13^{27} \bmod 55=7$.

Для перевірки підпису обраховуємо $w=7^3 \bmod 55=13$. Оскільки це значення співпадає зі значенням геш-функції, підпис справжній.

Цифровий RSA-підпис має два суттєвих недоліки:

1. Він є дуже великим, а іноді необхідно, щоб підпис займав мало місця.
2. Генерація RSA-підпису - надто дорога операція, що унеможливлює його використання в окремих додатках.

Цифровий підпис на основі стандарту DSA

В DSA використовуються наступні параметри домена:

1. p - просте число p, де $2L^{-1} < p < 2L$, 512 =< L =< 1024 и L кратно 64;
2. q - простий дільник p-1, при цьому 2159 < q < 2160;
3. g = h(p-1)/q mod p, де h - будь-яке ціле число 1 < h < p - 1 таке, що h(p- 1)/q mod p > 1;
4. x - випадкове ціле число, 0 < x < q; 5. y = gx mod p;
5. k - випадкове ціле число, 0 < k < q.

Генерація підпису виконується наступним способом:

6. $r = (g^k \bmod p) \bmod q$
7. $s = (k^{-1}(SHA(M) + xr)) \bmod q$.
8. Якщо r = 0 або s = 0, має бути згенеровано нове k і обчислений новий підпис.

Тут SHA (M) - 160-бітний бінарний рядок, що отримується в наслідок застосування геширування за алгоритмом SHA-1.

Перевірка підпису:

Числа p, q, g і відкритий ключ знаходяться у відкритому доступі. Нехай M', r' і s' отримані версії M, r і s, відповідно, і нехай y - відкритий ключ. При перевірці підпису спочатку потрібно подивитися, чи виконуються наступні нерівності: 0 <r '<q, 0 <s '<q. Якщо

хоча б одне нерівність не виконано, підпис повинна бути відкинута. Якщо умови нерівностей виконані, виробляються наступні обчислення:

1. $w = (s')^{-1} \bmod q$
2. $u1 = ((SHA(M'))\ w) \bmod q$
3. $u2 = ((r')\ w) \bmod q$
4. $v = (((g)^{u1}\ (y)^{u2}) \bmod p) \bmod q$.

Якщо v = r ', то справжність підпису підтверджена.

Приклад. Виберем такі параметри домену: q=13, p=4q+1=53 і $g=2^4 \bmod 53=16$. Нехай ключова пара користувача приймає значення x=3, $y=16^3 \bmod 53 =15$.

Будемо вважати, що геш-функція повідомлення SHA(M)=5. В якості ефемерного ключа виберемо k=2.

Генеруємо підпис: $r = (16^2 \bmod 53) \bmod 13 =5$, $s = ((5 + 3*5)/2) \bmod 13=10$.

Перевірка підпису: $w = (10)^{-1} \bmod 13 = 4$, $u1 = (5*4) \bmod 13=7$, $u2 = (5*4) \bmod 13=7$, $v = (16^7*15^7) \bmod 53) \bmod 13=5$.

На платформі .NET цифровий підпис RSA реалізується за допомогою об'єктів класу ***RSACryptoServiceProvider*** з простору імен System.Security.Cryptography.

Порядок створення цифрового підпису полягає у такому:

Створюється контейнер з ключами за замовчуванням:

```
CspParameters signParam = new CspParameters(); signParam.KeyContainerName = "Bob";
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(signParam);
//Для контролю можна виввести згенерований секретний ключ
//Console.WriteLine("---Secret key:\n{0}\n", rsa.ToXmlString(false));
```

*Публічний ключ експортується для передачі іншій стороні в XML- формат*і:

```
string pubKey = rsa.ToXmlString(false);
Console.WriteLine("---Public key:\n{0}\n", pubKey);
```

Вхідне повідомлення представляється у вигляді байтової послідовності:

```
byte[] message = Encoding.UTF8.GetBytes("Hello world!");
//Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));
```

Створюється дайджест повідомлення за алгоритмом SHA-1:

```
SHA1 sha1 = new SHA1CryptoServiceProvider();
byte[] hmessage = sha1.ComputeHash(message);
//Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
```

Створюється підпис для дайджесту, наприклад, в 16-му представленні:

```
byte[] signature = rsa.SignHash(hmessage, "sha1");
Console.WriteLine("---Signature: \n{0}\n", BitConverter.ToString(signature));
```

Підпис додається до повідомлення.

Порядок перевірки цифрового підпису полягає у такому:

Вхідне повідомлення представляється у вигляді байтової послідовності:

```
byte[] message = Encoding.UTF8.GetBytes("Hello world!");
Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));
```

Створюється дайджест повідомлення за алгоритмом SHA-1:

```
SHA1 sha1 = new SHA1CryptoServiceProvider();
byte[] hmessage sha1.ComputeHash(message); Console.WriteLine("---Hesh: \n{0}\n",
BitConverter.ToString(hmessage));
```

Створюється об'єкт RSA, до якого експортується публічний ключ з контейнера:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(); string
pubKey =
"<RSAKeyValue><Modulus>mJ32CjZjzD2Qw4oRc/3O4xyVBLLPHeXELhIa9kwPhPCERkhuvR
OalKgfodSgOrVr2K9TgTAMw3Ua4Q9O/vWhJRzsQPcSEYc5wJuHU9QY0a9jPn7WJQY91uCwf
A54GsVYTLo2VI60DEtuES7Dhlj4VK5KemvudEkmHiY8/BfO0XM=</Modulus><Exponent>AQ
AB</Exponent><P>z9JDG1Avu9q5LlBNG6NQcxdhZc+HnJsQahjmza/fZb6T/K0tThAjAM18dH2
```

gCCac05oJn2QEyCbtW5RgQ3lgvQ==</P><Q>u/9yCtIK1GjfS16K5pNkSqdVZr2QDHbjy0cHiO7
LfKBtWugUyN3FCSSUBooF2DmNFvRKDm1mZ4iPP7nMsqYV7w==</Q><DP>burHyjIX5+kq
4J8XKGMXsvWNlCsZM+pG7nlv5eOyFbmLflZnUbynEeyaOgo6eV8yYHVcIWfebXzpPfGJFzo
W+Q==</DP><DQ>hZeph6zoyzZW7u0ZEW7NxwsP8flk4qadizdHUHQ4n7A05XOkSXTmbm/S
zK7KJnQHIbeo5IWzToFJIkS7BHxnew==</DQ><InverseQ>jbhzN1BGEp4rA8njVFoHv8nYoyw
X0fG0tVoeMkuu+V5St81pD5oY5rz+OMksTxD+scpSF8DnhEewtBPMDOTJDg==</InverseQ><
D>P2u7NWBT0RePgPIEO1t5c7g1h0AGKp8hbCcZ7Ff2ZyhOxuqeQQGfrQGwRc8pmjxsg/ZoZu4
Ehk93DyiVSz53AfGe7vW+Mwk6jM71deoSyBdKpP1S/cpssOJHY781Tcx8jqqmP/gN19jEUkU7
mJcmv4ahkis79THvo/F vAmkJE=</D></RSAKeyValue>"; rsa.FromXmlString(pubKey);

Вхідне повідомлення представляється у вигляді байтової послідовності:

byte[] message = Encoding.UTF8.GetBytes("Hello world!"); Console.WriteLine("---
HexMessage: \n{0}\n ", BitConverter.ToString(message));

Перевіряється підпис для дайджесту:

string hexSign
="4725D1135F715C2FCFA2C6E93C839B106F1F8B2481CD599A5062652C39D2B63675D5254
377B7400AF85E3338F8023AA53D59AEC0144815C76FD02E22C0F3D8B976CDCDCF1DE42B
AF7D4314C3124B54E4B17B114A6E226001913AE15939584001AA1E98FA81C1F435F0F272D
FEED1C27476B6F2C3E5A7AF968CAC149892B7A198";

byte[] signature = StringToByteArray(hexSign);

bool match = rsa.VerifyHash(hmessage, "sha1", signature);
Console.WriteLine("Verdict:\n{0}\n", match);

Тут перетворення 16-вого цифрового підпису в байтовий масив
використовується функція:

public static byte[] StringToByteArray(string hex)
{ return Enumerable.Range(0, hex.Length)
.Where(x => x % 2 == 0)
.Select(x => Convert.ToByte(hex.Substring(x, 2), 16)) .ToArray();
}

Можна обійтись без перетворення 16-вого цифрового підпису в байтовий масив. Для цього цифровий підпис треба зберегти, наприклад, у форматі Base64:

byte[] signature = rsa.SignHash(hmessage, "sha1");
Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

Тоді перевірити підпис можна так:

string base64Sign =
"RyXRE19xXC/PosbpPIObEG8fiySBzVmaUGJlLDnStjZ11SVDd7dACvheMzj4AjqlPVmuwBRI
Fcdv0C4iwPPYuXbNzc8d5CuvfUMUwxJLVOSxexFKbiJgAZE64Vk5WEABqh6Y+oHB9DXw8
nLf7tHCdHa28sPlp6+Wj KwUmJK3oZg="; byte[] signature =
Convert.FromBase64String(base64Sign);

Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ:

Завдання 1 Опис механізму програмної реалізації цифрового підпису

Підготувати короткий опис щодо програмних механізмів реалізації цифрового підпису які реалізовані в обраній мові програмування.

Завдання 2 Програмна реалізація накладання цифрового підпису

а) На обраній мові програмування здійснити програмну реалізацію накладання цифрового підпису. (в якості ключів для формування цифрового підпису можна використовувати як згенеровані програмно, так і існуючий акр кредитованих центрів формування цифрового підпису. На вибір студента)

b) Здійснити підписання електронного документу.

c) Здійснити перевірку накладеного цифрового підпису.

Оформити звіт.

Додаток 1. Лістинг програми зі створення і перевірки ЕЦП RSA.

```
using System;
using System.Text;
using System.Security.Cryptography;
using System.Linq;
namespace RSA_sign
{
class Program
{
//Функція створення підпису
static void GetSign(){
Console.WriteLine("\n<<<GetSign>>>:\n");
//Створюється контейнер з ключами за замовчуванням
CspParameters signParam = new CspParameters();
signParam.KeyContainerName = "Bob";
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(signParam);
//Для контролю виводиться згенерований секретний ключ
Console.WriteLine("---Secret key:\n{0}\n", rsa.ToXmlString(false));
//Публічний ключ експортується для передачі іншій стороні
string pubKey = rsa.ToXmlString(false);
Console.WriteLine("---Public key:\n{0}\n", pubKey);
//Вхідне повідомлення представляється у вигляді байтової послідовності
byte[] message = Encoding.UTF8.GetBytes("Hello world!");
Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));
//Створюється дайджест повідомлення за алгоритмом SHA-1
SHA1 sha1 = new SHA1CryptoServiceProvider();
byte[] hmessage = sha1.ComputeHash(message);
Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
//Створюється підпис для дайджесту
byte[] signature = rsa.SignHash(hmessage, "sha1");
        Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature)); }
//Функція перевірки підпису
static void VerifySign(){
Console.WriteLine("\n<<<VerifySign>>>:\n");
//Вхідне повідомлення представляється у вигляді байтової послідовності
byte[] message = Encoding.UTF8.GetBytes("Hello world!");
Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));
//Створюється дайджест повідомлення за алгоритмом SHA-1
SHA1 sha1 = new SHA1CryptoServiceProvider();
byte[] hmessage = sha1.ComputeHash(message);
Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
//Створюється об’єкт RSA з розміщенням ключів у контейнері
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
string pubKey =
"<RSAKeyValue><Modulus>mJ32CjZjzD2Qw4oRc/3O4xyVBLLPHeXELhIa9kwPhPCERkhuvR
OalKgfodSgOr
Vr2K9TgTAMw3Ua4Q9O/vWhJRzsQPcSEYc5wJuHU9QY0a9jPn7WJQY91uCwfA54GsVYTLo
2VI60DEtuES7Dhlj4V
K5KemvudEkmHiY8/BfO0XM=</Modulus><Exponent>AQAB</Exponent><P>z9JDG1Avu9q5
LlBNG6NQcxdhZ
c+HnJsQahjmza/fZb6T/K0tThAjAM18dH2gCCac05oJn2QEyCbtW5RgQ3lgvQ==</P><Q>u/9yCt
IK1GjfS16K
```

```
5pNkSqdVZr2QDHbjy0cHiO7LfKBtWugUyN3FCSSUBooF2DmNFvRKDm1mZ4iPP7nMsqYV7
w==</Q><DP>burHyj
IX5+kq4J8XKGMXsvWNlCsZM+pG7nlv5eOyFbmLflZnUbynEeyaOgo6eV8yYHVcIWfebXzpPf
GJFzoW+Q==</DP
><DQ>hZeph6zoyzZW7u0ZEW7NxwsP8flk4qadizdHUHQ4n7A05XOkSXTmbm/SzK7KJnQHIb
eo5IWzToFJIkS7B
Hxnew==</DQ><InverseQ>jbhzN1BGEp4rA8njVFoHv8nYoywX0fG0tVoeMkuu+V5St81pD5oY
5rz+OMksTxD+
scpSF8DnhEewtBPMDOTJDg==</InverseQ><D>P2uU7NWBT0RePgPIEO1t5c7g1h0AGKp8hb
CcZ7Ff2ZyhOxuq
eQQGfrQGwRc8pmjxsg/ZoZu4Ehk93DyiVSz5k3AfGe7vW+Mwk6jM71deoSyBdKpP1S/cpssOJ
HY781Tcx8jqqm P/gN19jEUkU7mJcmv4ahkis79THvo/FvAmkJE=</D></RSAKeyValue>";
        rsa.FromXmlString(pubKey);
        //Перевіряється підпис для дайджесту
        String base64Sign =
"RyXRE19xXC/PosbpPIObEG8fiySBzVmaUGJlLDnStjZ11SVDd7dACvheMzj4AjqlPVmuwBRI
Fcdv0C4iwPPY u
        XbNzc8d5CuvfUMUwxJLVOSxexFKbiJgAZE64Vk5WEABqh6Y+oHB9DXw8nLf7tHCdH
a28sPlp6+WjKwUmJK3oZg =;
        byte[] signature = Convert.FromBase64String(base64Sign);
        Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));
        bool match = rsa.VerifyHash(hmessage, "sha1", signature);
        Console.WriteLine("---Verdict:\n{0}\n", match);
        }
        static void Main()
        {
        GetSign();
        VerifySign();
        }
        }
        }
```

Додаток 2. Лістинг програми зі створення і перевірки ЕЦП DSA.

```
using System; using System.Text;
using System.Security.Cryptography;
using System.Linq;
namespace DSA_sign
{
class Program
{
//Функція створення підпису
static void GetSign()
{
Console.WriteLine("\n<<<GetSign>>>:\n");
//Створюється об’єкт DSA з ключами за замовчуванням
DSACryptoServiceProvider dsa = new DSACryptoServiceProvider();
//Для контролю виводиться згенерований секретний ключ
string prKey = dsa.ToXmlString(true);
Console.WriteLine("---Secret key:\n{0}\n", dsa.ToXmlString(true));
//Публічний ключ експортується для передачі іншій стороні
string pubKey = dsa.ToXmlString(false);
Console.WriteLine("---Public key:\n{0}\n", pubKey);
//Вхідне повідомлення представляється у вигляді байтової послідовності
byte[] message = Encoding.UTF8.GetBytes("Hello world!");
```

```
        Console.WriteLine("---HexMessage: \n{0}\n", BitConverter.ToString(message));
        //Створюється дайджест повідомлення за алгоритмом SHA-1 SHA1
        sha1 = new SHA1CryptoServiceProvider();
        byte[] hmessage = sha1.ComputeHash(message);
        Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
        //Створюється підпис для дайджесту
        byte[] signature = dsa.SignHash(hmessage, "sha1");
        Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature)); }
        //Функція перевірки підпису
        static void VerifySign() {
        Console.WriteLine("\n<<<VerifySign>>>:\n");
        //Вхідне повідомлення представляється у вигляді байтової послідовності
        byte[] message = Encoding.UTF8.GetBytes("Hello world!");
        Console.WriteLine("---HexMessage: \n{0}\n ", BitConverter.ToString(message));
        //Створюється дайджест повідомлення за алгоритмом SHA-1
        SHA1 sha1 = new SHA1CryptoServiceProvider();
        byte[] hmessage = sha1.ComputeHash(message);
        Console.WriteLine("---Hesh: \n{0}\n", BitConverter.ToString(hmessage));
                //Створюється об’єкт DSA, до якого експортується публічний ключ з
            контейнера DSACryptoServiceProvider dsa = new DSACryptoServiceProvider(); string
            pubKey
        "<DSAKeyValue><P>jmQh1u62F7h3RMkRs+uhoGn4fQMhih8GyZPWmJJHRwcOVxmJC
5d5cR0k7PFf0x0+PvkHr
+wFSTvB2ZregiasXJ8WnHUvEcFCHbJ+iRiKQD1Rd75HfN8o8ViqL36Ia6PM1ZwI2Fqyc0zJsho
Zqg2CQX8cDwO7T/Ogea+S+5bnrrE=</P><Q>68r+49IOAxNZNJkG43A5I+Ma9hE=</Q><G>g9
RWkaYfOs1tOewwxFHZFgN9NFT
Dz6FW5UWN2bakbfyOOZ2xeveYQ87nJEg5nXv4ZSTo7mxP2AhuL8lFKKmGcPLDgPFwjcWqT
AeMTd3x6zKDb1Ev0
N4VTzzHD0GNLfTfdtQpTYuzzifp+gh2rtkmqF29g8DqOgG1uI9pVYfvYF0=</G><Y>Q6SrbyDo
6/T0n8aZrWow
e3YJKgEzEZhT7qAY8nZ2O7CvCIM7Y3M8/9DkddrAosk1X4pSye5bmBTERpDB1+I7TlAqS+s
OeUx69UL6OJlRGT
fdai51g9T/qy7nPqwgY/jBQx4UvPjLIsFaovCjjRDLTje9X8Q2i+5c/OSdHN8/mfQ=</Y><J>mpgE
UdSn4XqlW
ugO+TKaF4r7WFQkux4wRPRIJHoL/wBgjIY8oA9Ji8RgVOGgb9TI61ll5BTM4mfR/ucCgMHtoK
Rbi8dOzOgoj6h
dT1BWDg4vW23t6v8Up3/APn/tO+ZAOaL26rZuscyaJ1Ow</J><Seed>U1+1KTe5EGYeuLYQoj
zYeoTfGnk=</S eed><PgenCounter>Ag==</PgenCounter></DSAKeyValue>";
dsa.FromXmlString(pubKey);

        //Перевіряється підпис для дайджесту string
        base64Sign = "mxdOkPaZrDdDnl15sEL8GQim6B4/c2fbd2ksc4ByqxOVS7TA1ouXqg==";
        byte[] signature = Convert.FromBase64String(base64Sign);
        //Console.WriteLine("---Signature: \n{0}\n", Convert.ToBase64String(signature));
        bool match = dsa.VerifyHash(hmessage, "sha1", signature); Console.WriteLine("---
Verdict:\n{0}\n", match);

        static void Main()
        {
        //GetSign();
        VerifySign();
        }
        }
        }
```

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Nigel Cawthorne. Alan Turing: The Enigma Man. – Acturus, 2019. – 128 p.
1. Вишня В. Б. Основи інформаційної безпеки : навч. посібник / В. Б. Вишня, О. С. Гавриш, Е. В. Рижков. Дніпро : Дніпроп. держ.ун-т внутріш. справ, 2020. 128 с.
2. Гребенюк А.М. Основи управління інформаційною безпекою: навч. посібник / А.М. Гребенюк, Л.В. Рибальченко. Дніпро: Дніпроп. держ. унт внутріш. справ, 2020. – 144 с.
3. Інформаційна безпека. Підручник / В. В. Остроухов, М. М. Присяжнюк, О. І. Фармагей, М. М. Чеховська та ін.; під ред. В. В. Остроухова – К.: Видавництво Ліра-К, 2021. – 412 с.
4. Інформаційна безпека/ За ред. Ю. Я. Бобала та І. В. Горбатого, Львівська політехніка, 2019.-540 с.
2. Кібербезпека : сучасні технології захисту. Навчальний посібник для студентів вищих навчальних закладів. / С. Е. Остапов, С. П. Євсеєв,О.Г. Король. – Львів: «Новий Світ-2000», 2020 . – 678 с.
3. Кібербезпека в сучасному світі : матеріали ІІІ Всеукраїнської науково практичної конференції (м. Одеса, 19 листопада 2021 р.) / за ред. О. В. Дикого ;уклад.: С. А. Горбаченко, Н. І. Логінова. – Одеса, 2020. – 148 с.
4. Кібербезпека: лабораторний практикум з основ криптографічного захисту/ Євсеєв С.П. , Король О.Г. Новий світ-2000, 2021.-241 с.
5. Криптоаналіз. Криптографічні протоколи / О.М. Гапак // Навчальний посібник з курсу «Комп'ютерна криптографія» для студентів інженерно-технічного факультету спеціальності 123-«Комп’ютерна інженерія». Ужгород: видавництво ПП «АУТДОР-ШАРК», 2021р. – 96с..
5. Лісовська Ю. Кібербезпека. Ризики та заходи. - К.: Кондор, 2019. - 272 с.
6. Логінова Н. І. Правовий захист інформації : навч. посібн. / Н. І. Логінова, Р. Р. Дробожур. - Одеса : Фенікс, 2015. - 264 с.
7. Організаційно-правові основи захисту службової 0-64 інформації: навч. посіб. /1. П. Касперський, С. 0. Князєв, О. І. Матяш та ін. - Київ : Нац. акад. СБУ, 2017.-120 с.
8. Організація захисту інформації з обмеженим доступом: навч. посіб. /А.М.Гуз, І.П.Касперський, С.О.Князєв та ін. – К.: Нац. акад., СБУ, 2018. –252 с
6. Стандарти захисту персональних даних в соціальній сфері / М. В. Бем,, І. М. Городиський. –Львів: б.в., 2018. - 110 с.
7. Тарнавський Ю.А. Технології захисту інформації [Електронний ресурс]: підручник. – К.: КПІ ім. Ігоря Сікорського, 2018. – 162 с. Режим доступу до ресурсу: https://ela.kpi.ua/bitstream/123456789/23896/1/TZI_book.pdf