

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

Основи захисту інформації

*Методичні вказівки до виконання лабораторних робіт
для студентів денної форми навчання
за спеціальностями 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”*

ЗАТВЕРДЖЕНО

на засіданні кафедри кібербезпеки та
програмного забезпечення, протокол № 1
від 05.07.2017

Кропивницький

2017

Основи захисту інформації: методичні вказівки до виконання лабораторних робіт для студентів за спеціальностями 123 “Комп’ютерна інженерія”, 122 “Комп’ютерні науки”/ М-во освіти і науки України, Центральноукр. нац. техн. ун-т; [уклад. О. А. Смірнов, Є.В. Мелешко, О. К. Конопліцька-Слободенюк, В.Д. Хох, С.А. Смірнов] – Кропивницький: ЦНТУ, 2017. – 53 с.

Укладачі: Смірнов О. А., докт. техн. наук, професор;
Мелешко Є.В., канд. техн. наук., доцент
Конопліцька-Слободенюк О. К., викладач;
Хох В.Д., аспірант;
Смірнов С.А., канд. техн. наук, ст. викладач.

Рецензенти: Сидоренко В. В., докт. техн. наук, професор;
Доренський О. П., канд. техн. наук.

© Центральноукраїнський
національний технічний
університет, 2017

3MICT

ВСТУП.....	4
Лабораторна робота №1 Закони України стосовно захисту інформації	8
Лабораторна робота №2. Симетричні алгоритми шифрування. DES.....	9
Лабораторна робота №3. Асиметричні алгоритми шифрування. RSA	17
Лабораторна робота №4 Генератори псевдовипадкових чисел.....	20
Лабораторна робота №5. Геш-функції. MD5 та SHA-1.....	22
Лабораторна робота №6. Автентифікація даних і електронний цифровий підпис	30
Лабораторна робота №7. Криптоаналіз. Взлам RSA	34
Лабораторна робота №8. Програмні закладки. Клавіатурний шпигун	37
Лабораторна робота №9. Стеганографія	44
Список використаної літератури.....	50

ВСТУП

Мета: Основна мета курсу полягає в отриманні досконалих знань у області захисту інформації а також отримання студентами навичок створення і розміщення в мережі проектів з захисту інформації.

Завдання:

Вивчення теоретичних основ і положень захисту інформації.

Вивчення способів криптографічного перетворення інформації.

Отримання необхідних теоретичних знань побудови систем захисту інформації.

Отримання практичних навиків адміністрування систем захисту інформації.

У результаті вивчення навчальної дисципліни студент повинен:

– знати:

- основні визначення та положення теорії захисту інформації;
- нормативно-правову базу щодо захисту інформації в Україні.
- стандарти у галузі захисту інформації;
- симетричну криптографію;
- несиметричну (асиметричну, з відкритим ключем) криптографію;
- генератори випадкових чисел;
- автентифікацію;
- геш-функції й коди автентифікації повідомлень – MAC;
- цифровий підпис;
- розподіл ключів;
- технічні засоби знімання й захисту інформації;
- криптоаналіз;
- введення в мережну безпеку Інтернет;
- віруси й антивіруси;
- адміністративний рівень інформаційної безпеки;
- стеганографію;

- біометричні методи захисту інформації;
- безпеку операційних систем.

– **Вміти:** Програмно реалізовувати наступні проекти:

- база даних щодо законів України стосовно захисту інформації;
- симетричні алгоритми шифрування. DES.;
- асиметричні алгоритми шифрування. RSA.;
- генератори псевдовипадкових чисел. Геш-функції. MD5 та SHA-1.;
- автентифікація даних і електронний цифровий підпис;
- криптоаналіз. Взлам RSA.;
- програмні закладки;
- клавіатурний шпигун.;
- стеганографія.

Структурно логічна схема підготовки магістра.

Враховуючи послідовність накопичення знань та інформації, дисципліна вивчається після викладання наступних дисциплін:

- Вища математика. Теорія ймовірності та математична статистика.
- Алгоритми та методи обчислень.
- Програмування.
- Організація баз даних.
- Інженерія програмного забезпечення.
- Системне програмне забезпечення.

Форма підсумкового контролю: іспит.

Максимальну кількість балів студент може одержати у випадку відвідування всіх лекцій, лабораторних занять, виконання і захисту виконаних самостійних завдань у встановлений термін, проходження контролю.

В якості самостійного завдання необхідно виконати теоретичну роботу, згідно обраної студентом теми.

Шкала оцінювання: національна та ECTS

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою
		для екзамену
90 – 100	A	відмінно
82-89	B	добре
74-81	C	
64-73	D	задовільно
60-63	E	
35-59	FX	незадовільно з можливістю повторного складання
0-34	F	незадовільно з обов'язковим повторним вивченням дисципліни

Вибравши предметну область, над якою ви будете працювати, ви повинні виконати завдання до лабораторних робіт, а також відповісти на питання в кінці кожної лабораторної роботи. Звіт повинен містити хід виконання завдань, а також графічні матеріали, що підтверджують виконання цих завдань.

Самостійна робота

№ з/п	Назва теми	Кількість годин
		денна форма навчання
1	Тема 1. Основні визначення та положення теорії захисту інформації.	3
2	Тема 2. Нормативно-правова база щодо захисту інформації в Україні. Стандарти у галузі захисту інформації	4
3	Тема 3. Симетрична криптографія	3

4	Тема 4. Несиметрична (асиметрична, з відкритим ключем) криптографія	4
5	Тема 5. Генератори випадкових чисел	3
6	Тема 6. Автентифікація	4
7	Тема 7. Геш-функції й коди автентифікації повідомлень – MAC	3
8	Тема 8. Цифровий підпис	4
9	Тема 9. Розподіл ключів	3
10	Тема 10. Технічні засоби знімання й захисту інформації	4
11	Тема 11. Криптоаналіз	3
12	Тема 12. Введення в мережну безпеку Інтернет. Віруси й антивіруси	4
13	Тема 13. Адміністративний рівень інформаційної безпеки	3
14	Тема 14. Стеганографія	3
Усього годин		48

Для опанування матеріалу дисципліни «Основи захисту інформації» окрім лекційних та лабораторних занять, тобто аудиторного навантаження, значна увага приділяється самостійній роботі.

До основних видів самостійної роботи студента відносимо:

1. Вивчення лекційного матеріалу.
2. Робота з літературними джерелами.
3. Розв’язання практичних задач за індивідуальними варіантами.
4. Підготовка до модульних, підсумкового контролю, екзамену (денна та заочна).
5. Виконання курсової роботи для денної форми навчання.

Для підвищення рейтингу впродовж семестру студент може виконати згідно запропонованої викладачем теми самостійну роботу, обсяг якої складає не менше 10 сторінок.

7. Індивідуальні завдання

Лабораторна робота № 1

Тема: Закони України стосовно захисту інформації

Теоретичні відомості.

Посилання на закони та стандарти дивіться в лекціях №2-3.

Закони можна переглянути на сайті Верховної Ради України (<http://portal.rada.gov.ua/>) та Державної служби спецзв'язку (<http://www.dstszi.gov.ua/>).

Завдання: Створити базу даних по законам України та стандартам, що стосуються захисту інформації. Базу реалізовувати на будь-якій мові програмування з використанням довільної бази даних.

Лабораторна робота № 2

Тема: Симетричні алгоритми шифрування. DES.

Теоретичні відомості.

Розглянемо загальну схему симетричної, або традиційної, криптографії.

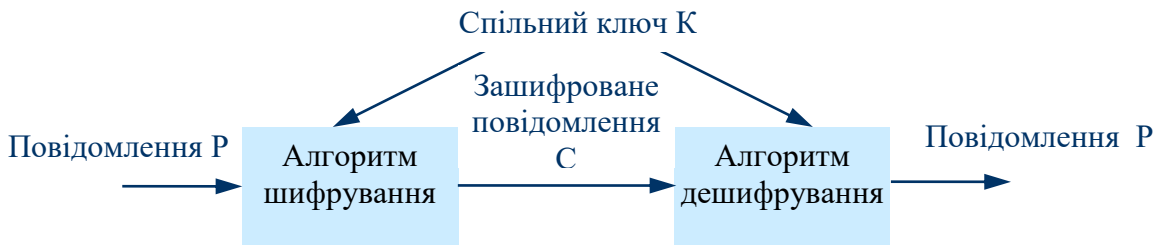


Рис. 1. Загальна схема симетричного шифрування

В симетричному шифруванні використовується один секретний ключ як для шифрування так і для дешифрування (або по одному ключі можна легко обчислити інший).

До симетричного шифрування відносяться такі алгоритми як DES, Triple-DES, Blowfish, IDEA, RC4 тощо. Багато розповсюджених шифрів (наприклад IDEA, DES, BLOWFISH) є блоковими шифрами. Це означає, що вони беруть блоки даних фіксованого розміру (звичайно 64 біта), і перетворюють їх в інший 64-бітний блок за допомогою функції, що задається ключем.

Найпоширенішим і найбільш відомим алгоритмом симетричного шифрування є **DES** (Data Encryption Standard). Алгоритм був розроблений в 1977 році, в 1980 році був прийнятий NIST (National Institute of Standards and Technology США) як стандарт (FIPS PUB 46).

Реалізації DES можуть бути знайдені, наприклад, у бібліотеках libdes, alodes, SSLeay, Crypto++, descore, chalmers-des і в destoo.

Існують розробки, що дозволяють виконувати шифрування в рамках стандарту DES апаратним способом, що забезпечує досить високу швидкодію.

Алгоритм шифрування DES.

Дані шифруються 64-бітовими блоками, використовуючи 56-бітовий ключ. Ключ шифрування повинен бути відомий як відправнику так і одержувачу.

Процес шифрування складається із таких етапів:

- 1) початкова перестановка 64-бітного вихідного тексту, під час якої біти переставляються у відповідності зі стандартною таблицею.
- 2) 16 раундів однієї й тієї ж функції, що використовує операції зсуву і підстановки.
- 3) ліва і права половини виходу останньої (16-ї) ітерації міняються місцями.
- 4) перестановка результату, отриманого на третьому етапі, ця перестановка інверсна початковій перестановці.

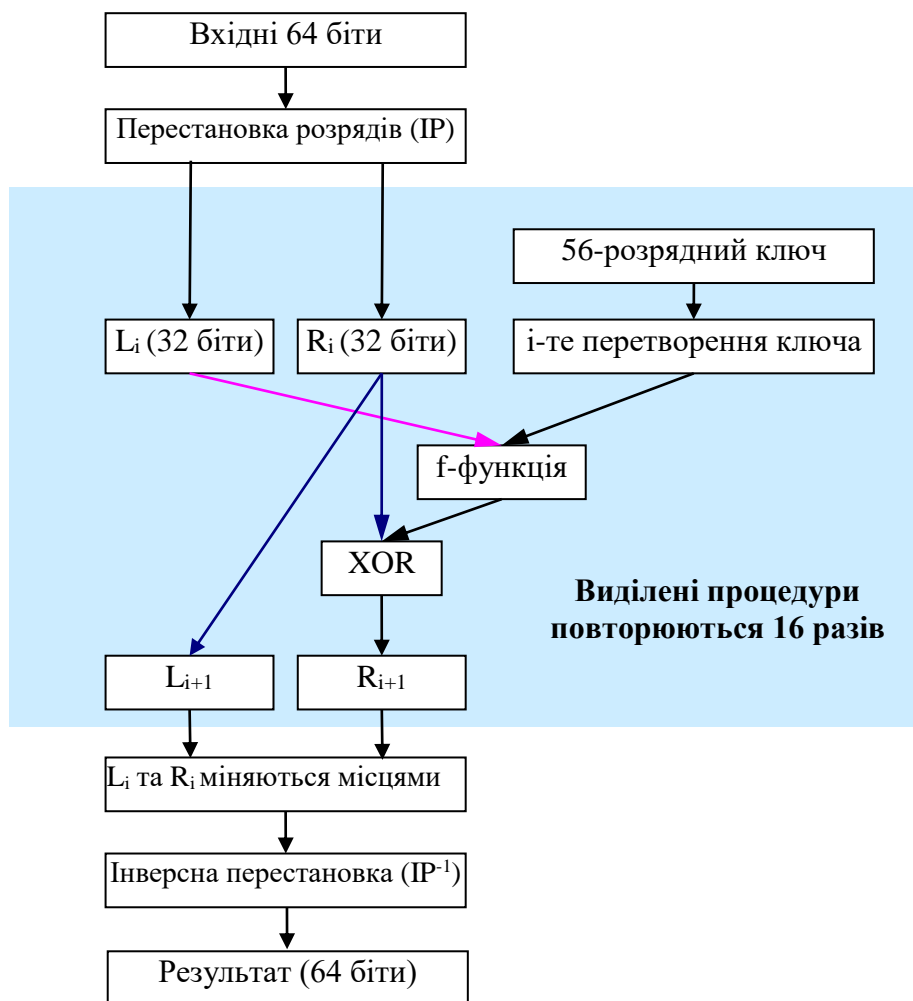


Рис. 2. Структурна схема реалізації алгоритму DES

Тепер більш детально розглянемо наведені вище етапи алгоритму.

1 Етап

Шифрування починається з перестановки бітів (IP – Initial Permutation) в 64-розрядному блоці вихідних даних. 58-ой біт стає першим, 50-ий – другим і т.д.

Схема початкової перестановки бітів (IP):

```
58 50 42 34 26 18 10 2
60 52 44 36 28 20 12 4
62 54 46 38 30 22 14 6
64 56 48 40 32 24 16 8
57 49 41 33 25 17 9 1
59 51 43 35 27 19 11 3
61 53 45 37 29 21 13 5
63 55 47 39 31 23 15 7
```

2 Етап

Отриманий блок ділиться на дві 32-розрядні частини L_0 і R_0 . Далі 16 разів повторюються наступні 4 процедури:

- 1) Перетворення ключа з урахуванням номера ітерації i (перестановки розрядів з видаленням 8 біт, у результаті виходить 48-розрядний ключ)
- 2) За допомогою функції $f(L_i, K_i)$ генерується 32-розрядний вихідний код.
- 3) Виконується операція XOR: $R_i \otimes f(L_i, K_i)$, результат позначається R_{i+1} .
- 4) Виконується операція присвоєння $L_{i+1} = R_i$.

Перетворення ключів K_n ($n = 1, \dots, 16$; $K_n = KS(n, \text{key})$, де n – номер ітерації) здійснюються відповідно до алгоритму, показаному на рис. 3.

Для описання алгоритму обчислення ключів K_n (функція KS) досить визначити структури “Вибір 1” та “Вибір 2”, а також задати схему зсувів вліво (табл. 3). “Вибір 1” та “Вибір 2” являють собою перестановки бітів ключа (PC-1 і PC-2; табл. 1, 2). При необхідності біти 8, 16, ..., 64 можуть використовуватися для контролю парності.

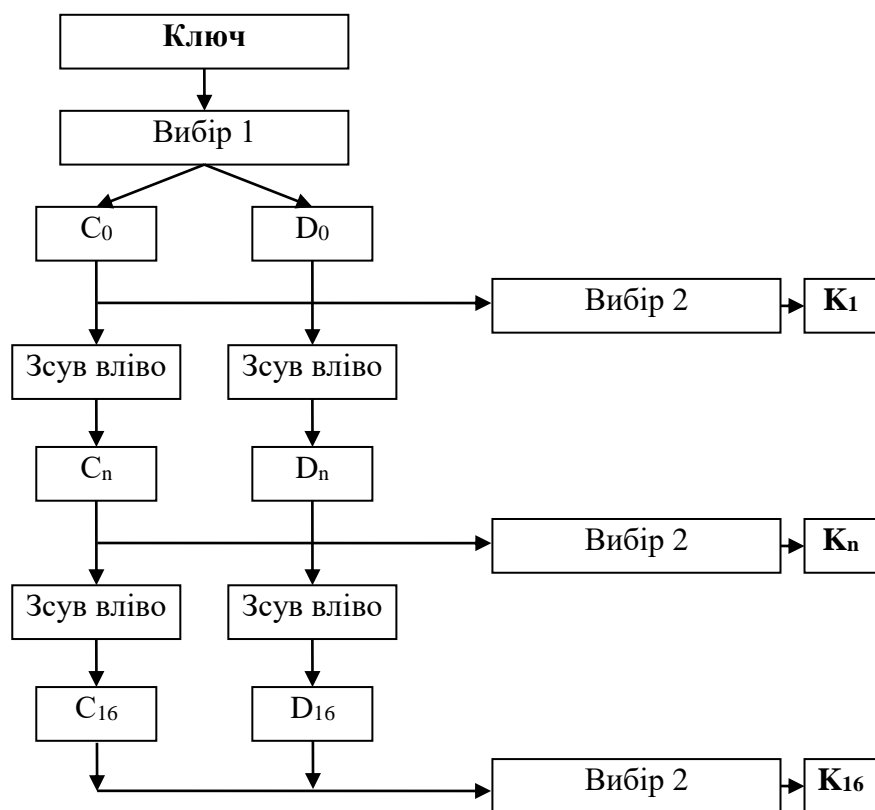


Рис. 3. Алгоритм обчислення послідовності ключів K_n

Таблиця 1. РС-1 (Вибір 1)

Заповнення С							Заповнення D						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

Таблиця 2. РС-2 (Вибір 2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Вибір бітів по таблицях з відповідних блоків здійснюється в такий спосіб. Таблиця розглядається як послідовність її рядків, записаних один за одним,

починаючи з першого рядка. Біти блоку даних відповідної довжини нумеруються зліва на право, починаючи з одиниці. Кожний елемент m таблиці розглядається, як номер біта b_m у блоці даних. Перетворення полягає в заміні всіх елементів m на біти b_s .

У C_0 увійдуть біти 57, 49, 41,..., 44 і 36, а в D_0 – 63, 55, 47,..., 12 і 4. Тому що схема зсувів задана (табл. 2) $C_1, D_1; C_n, D_n$ і так далі можуть бути легко отримані з C_0 і D_0 . Так, наприклад, C_3 і D_3 виходять із C_2 і D_2 циклічним зсувом вліво на 2 розряди.

Таблиця 3 Схема зсувів ключа

Номер циклу	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Зсув вліво (шифрування)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Зсув вправо (розшифрування)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Схема роботи функції f показана на рис. 4. Спочатку 32 вхідні розряди розширюються за допомогою перетворення EP до 48, при цьому деякі розряди повторюються.

Функція розширення EP:

```

32 1 2 3 4 5
4 5 6 7 8 9
8 9 10 11 12 13
12 13 14 15 16 17
16 17 18 19 20 21
20 21 22 23 24 25
24 25 26 27 28 29
28 29 30 31 32 1

```

Для отриманого 48-розрядного коду й ключа виконується операція XOR. Результуючий 48-розрядний код перетвориться в 32-розрядний за допомогою S-Матриць. На виході S-Матриць здійснюється перестановка за схемою Р-блока.

Р-блок:

16 7 20 21 29 12 28 17

1 15 23 26 5 18 31 10

2 8 24 14 32 27 3 9

19 13 30 6 22 11 4 25

S-Матриці являють собою таблиці, що містять 4-рядки і 16 стовпців.

S1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	32	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

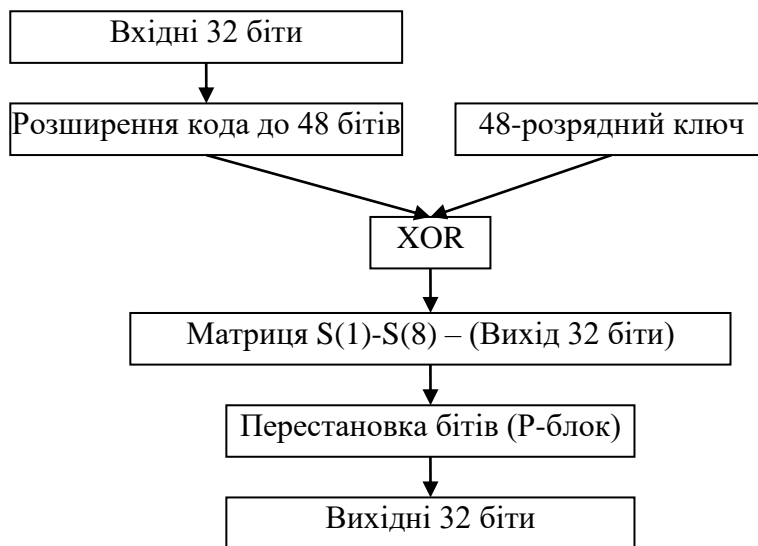


Рис. 4. Алгоритм роботи функції f

Вихідний 48-розрядний код ділиться на 8 груп по 6 розрядів. Перший і останній розряд у групі використовується як адреса рядка, а середні 4 розряди – як адреса стовпця. У результаті кожен 6 біт коду перетворюється в 4 біти, а весь 48-розрядний код в 32-розрядний, для цього потрібно 8 S-Матриць (*S-boxes*).

3 Етап

L_{16} та R_{16} міняються місцями.

4 Етап

Інверсна перестановка розрядів здійснюється наступним розміщенням 64 бітів зашифрованих даних (першим бітом стає 40-вий, другий 8-мим і т.д.).

Схема інверсної перестановки бітів (IP^{-1}):

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Алгоритм дешифрування DES

DES використовує для шифрування блока одну і ту ж функцію, різниця полягає лише в тому, що ключі повинні використовуватися в зворотному порядку (якщо для шифрування використовувалися ключі $K_1, K_2, K_3, K_4, K_5, \dots, K_{16}$, то ключами дешифровки будуть $K_{16}, K_{15}, K_{14}, K_{13}, K_{12}, \dots, K_1$). Це досягається тим, що зсув здійснюється не вліво, а вправо (дивись табл. 3).

Проблеми DES

Так як довжина ключа дорівнює 56 бітів, існує 256 можливих ключів. На сьогодні така довжина ключа недостатня, оскільки допускає успішне застосування лобових атак. Альтернативою DES можна вважати *потрійний DES*, IDEA, а також алгоритм Rijndael, прийнятий як новий стандарт на алгоритми симетричного шифрування.

Також без відповіді поки залишається питання, чи можливий криптоаналіз із використанням існуючих характеристик алгоритму *DES*. Основою алгоритму є вісім таблиць підстановки, або *S-boxes*, які застосовуються в кожній ітерації. Існує небезпека, що ці *S-boxes* конструювалися таким чином, що криптоаналіз можливий для зломщика, що знає слабкі місця *S-boxes*. Протягом багатьох років обговорювалося як стандартне, так і несподіване поведіння *S-boxes*, але все-таки нікому не вдалося виявити їх фатально слабкі місця.

Завдання:

- Створити програму, що реалізує алгоритм DES.
- Зашифрувати та розшифрувати файл об'ємом не менший 50 кб.

Лабораторна робота №3

Тема: Асиметричні алгоритми шифрування. RSA.

Теоретичні відомості.

Концепція криптосистеми з відкритим ключем

Ефективними системами криптографічного захисту даних є асиметричні криптосистеми (криптосистемами з відкритим ключем). У таких системах для шифрування даних використовується один ключ, а для розшифрування – інший (звідси й назва – асиметричні). Перший ключ є відкритим і може бути опублікований для використання всіма користувачами системи, які зашифровують дані. Розшифрування даних за допомогою відкритого ключа неможливо. Для розшифрування даних одержувач зашифрованої інформації використовує другий ключ, що є закритим (секретним). Зрозуміло, ключ розшифрування не може бути визначений із ключа шифрування. Узагальнена схема асиметричної криптосистеми з відкритим ключем показана на рис.1. Генератор ключів доцільно розташовувати на стороні одержувача В (щоб не пересилати секретний ключ КВ по незахищеному каналу).

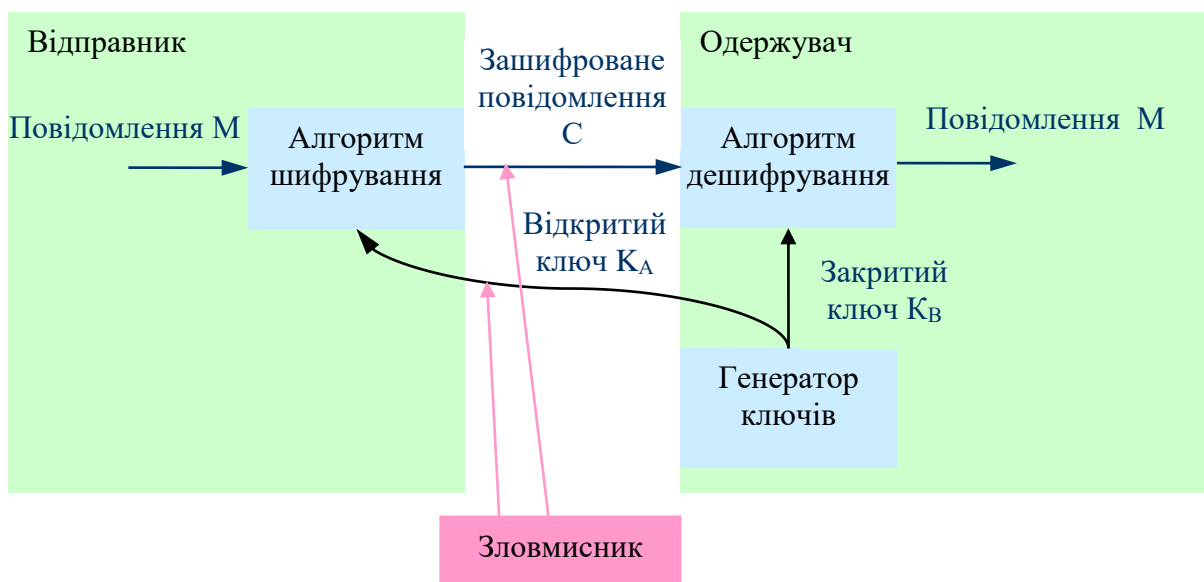


Рис.1. Узагальнена схема асиметричної криптосистеми з відкритим ключем.

Криптосистема шифрування даних RSA

Алгоритм RSA запропонували в 1978 р. три автори: Р.Райвест (Rivest), А.Шамир (Shamir) і А.Адлеман (Adleman). Алгоритм одержав свою назву по перших літерах їх прізвищ. Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, що може працювати як у режимі шифрування даних, так і в режимі електронного цифрового підпису.

Надійність алгоритму ґрунтується на важкості факторизації великих чисел та важкості обчислення дискретних логарифмів.

Алгоритм RSA складається з наступних пунктів:

1. Вибрати прості числа p і q .
2. Обчислити $n = p * q$.
3. Обчислити $m = (p - 1) * (q - 1)$.
4. Вибрати число d взаємно просте з m .
5. Вибрати число e таке, щоб $e * d = 1 \pmod{m}$.

Число d – закритий ключ, e – відкритий.

Повідомлення перед шифруванням необхідно розбити на блоки – числа від 0 до $n - 1$. Шифрування та дешифрування даних здійснюються в такий спосіб:

– Шифрування: $b = a^e \pmod{n}$

– Дешифрування: $a = b^d \pmod{n}$

Слід також зазначити, що ключі e і d рівноправні, тобто повідомлення можна шифрувати як ключем e , так і ключем d , при цьому розшифровка повинна бути зроблена за допомогою іншого ключа.

Знаходження взаємно простих чисел

На кроці 4 алгоритми RSA необхідно знайти число d взаємно просте з m . Число d повинне бути менше m , таким чином розрядність числа d дорівнює сумі бітів у числах p і q . Для знаходження взаємно простих чисел використовується алгоритм Евкліда, що знаходить найбільший спільний дільник двох чисел. Якщо знайдений дільник більше одиниці, то необхідно вибрати інше число d і повторити перевірку.

Алгоритм Евкліда

1. Беруться числа a і b
2. Обчислюється r – залишок від ділення a на b : $a = b * q + r$
3. Якщо $r = 0$, то b – шукане число (найбільший спільний дільник), кінець
4. Замінити пари чисел $\langle a, b \rangle$ парою $\langle b, r \rangle$, перейти до пункту 2

При обчисленні найбільшого загального дільника за допомогою алгоритму Евкліда буде виконано не більше $5 * p$ операцій ділення із залишком, де p є кількість цифр у десятковому записі меншого із чисел a і b . На практиці алгоритм працює дуже швидко.

В 5-му пункті алгоритму RSA передбачається знаходження такого числа e , щоб $e * d = 1 \pmod{m}$. Обчислення числа e зводиться до рішення рівняння $m * x + d * e = 1$ в натуральних числах. Число x не важливе. Для цього потрібно використати модифікований алгоритм Евкліда, який працює тільки якщо числа d та m взаємно прості.

Завдання:

- Реалізувати алгоритм RSA.
- Зашифрувати та розшифрувати файл об'ємом не менший 50 кб.
- Порівняти зі швидкістю виконання DES (лабораторна робота №2)

Лабораторна робота № 4

Тема: Генератори псевдовипадкових чисел

Теоретичні відомості

Криптографічні системи мають потребу в надійному генераторі випадкових чисел, значення якого не зможе передбачити атакуючий. Випадкові числа звичайно використовуються для генерації ключів сесії і їхня якість критичним образом впливає на якість системи в цілому. Генератор випадкових чисел легко випустити із уваги, і тоді він стає найслабшою ланкою системи.

Деякі машини можуть мати спеціальні апаратні генератори шуму. Шум від струму витоку діода або транзистора, молодші біти аудіосигнала, інтервали між перериваннями тощо. Усе це хороші джерела випадкових даних якщо їх пропустити через підходящу геш-функцію. Гарною ідеєю є одержання в міру можливості справжнього шуму з оточення.

Менш надійними, але простішими в реалізації є генератори псевдовипадкових чисел на основі математичних функцій.

Найпростіша послідовність, яку можна запропонувати для реалізації генератора рівномірного розподілу:

$$I_{(j+1)} = a * I_{(j)} \pmod{m}$$

При відповідному виборі констант. Константи запропоновані Park та Miller: $a = 75 = 16807$, $m = 2^{31}-1 = 2147483647$

і протестовані в дослідженнях Lewis, Goodman, Miller (1969).

Пряма реалізація цього методу можлива на мовах асемблера, але мови високого рівня можуть при цьому зафіксувати переповнення. Для обходу цього Schrage запропонував метод часткової факторизації модуля. Модуль розкладається таким чином:

$$m = a * q + r$$

Якщо $r < q$ і $0 < z < m-1$, то при цьому величини $a * (z \bmod q)$ і $r * [z/q]$ завжди лежать в інтервалі $0, \dots, m-1$. Для множення $(a * z) \pmod{m}$ при цьому використовується алгоритм:

$$t = a(z \bmod q) - r[z/q]$$

якщо $t < 0$, то $t + = m$.

$$(a * z) \bmod m = t.$$

У випадку констант Парка-Міллера можна використати $q = 12773$ і $r = 2836$.

Для асиметричних систем необхідно генерувати прості випадкові числа. Простий спосіб перевірки на простоту – ділення числа на всі числа менші за нього не працездатний уже з 32-бітними числами (потрібно дуже багато часу на виконання).

У цьому випадку, для вибору простих чисел використовують інші методи, один з яких представлений нижче.

Для перевірки числа на простоту розглянемо алгоритм Рабіна-Міллера, що робить наступне:

- 1) генерується випадкове число p , та встановлюється в одиницю його молодший біт;
- 2) обчислюється b – число дільників $(p - 1)$ на 2 (тобто 2^b – це найбільша ступінь числа 2, на яку ділиться $(p - 1)$);
- 3) обчислюється m , таке що $p = 1 + 2^b * m$;
- 4) вибирається випадкове число a , менше ніж p ;
- 5) встановлюється $j = 0$ та $z = a^m \bmod p$;
- 6) якщо $z = 1$ або якщо $z = p - 1$, то p **проходить перевірку** і може бути простим числом;
- 7) якщо $j > 0$ і $z = 1$, то p не є простим числом;
- 8) встановлюється $j = j + 1$. Якщо $j < b$ та $z \neq p - 1$, встановити $z = z^2$ та повернутися на етап (7). Якщо $z = p - 1$ то p проходить перевірку і може бути простим числом;
- 9) якщо $j = b$ та $z \neq p - 1$, то p не є простим числом.

Для прискорення швидкодії даного алгоритму, доцільно перед його використанням робити перевірку на те, чи не ділиться випадкове число p на 3, 5 та 7, це одразу дозволяє відсікати 54% непростих чисел, ще до етапу (7).

Завдання:

Створити програму, що генерує псевдовипадкові числа та перевіряє їх на простоту.

Лабораторна робота № 5

Тема: Геш-функції. MD5 та SHA-1.

Теоретичні відомості

Геш-функція – це така функція, що перетворює вхідний масив даних довільної довжини у вихідний бітовий рядок фіксованої довжини так, щоб зміна вхідних даних приводила до непередбачуваної зміни вихідних даних. Такі перетворення також називаються хешуванням або функціями згортки, а їх результати називають гешем, геш-кодом або дайджестом повідомлення.

В криптографії геш-функції використовуються у створенні електронного цифрового підпису та для автентифікації користувачів (коли в базі банних замість паролів містяться їх геш-значення).

Геш-функції також використовуються в деяких структурах даних – хеш таблицях і декартових деревах.

Криптографічна геш-функція повинна забезпечувати:

– стійкість до колізій: два різні набори даних повинні мати різні результати перетворення, тобто для заданого повідомлення M повинно бути практично неможливо підібрати інше повідомлення M' , що має такий же хеш.

– безповоротність (неможливість обчислити початкові дані по результату перетворення)

Згідно парадоксу про «дні народження», знаходження колізії для геш-функції з довжиною геш-коду n біт вимагає $O(\sqrt{n})$ операцій. Таким чином, цим числом оцінюється стійкість геш-функції до колізій.

Більшість геш-функцій будується на основі односпрямованої функції $f()$, яка створює вихідне значення довжиною n при наявності двох вхідних довжиною n . Цими вхідними значеннями являються блок тексту M , та геш-значення H_{i-1} попереднього блока тексту (рис.1).

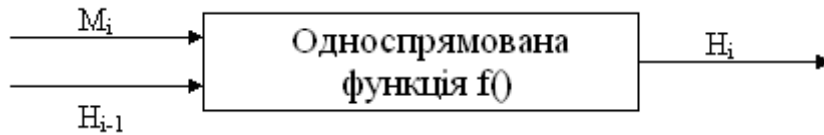


Рис.1. Побудова односпрямованої геш-функції

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

Геш-значення, що обчислюється при введенні останнього блоку тексту, стає геш-значенням усього повідомлення M .

У результаті односпрямована геш-функція завжди формує вихід фіксованої довжини n (незалежно від довжини вхідного тексту).

MD5 (Message Digest 5) – 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом у 1991 році.

Алгоритм MD5.

Константи, що використовуються в даному алгоритмі:

1. Початкові значення для ініціалізації змінних:

$$H[0] = 67452301_{16}$$

$$H[1] = EFCDAB89_{16}$$

$$H[2] = 98BADCFE_{16}$$

$$H[3] = 10325476_{16};$$

2. Константи додавання в раундах:

$$y[j] = \text{HIGHEST_32_BITS}(\text{ABS}(\text{SIN}(j+1))) \quad j = 0 \dots 63,$$

де функція $\text{HIGHEST_32_BITS}(X)$ визначає 32 самих старших біта з двійкового запису дробового числа X , а операнд $\text{SIN}(j+1)$ вважається взятим в радіанах.

3. Масив порядку вибору комірок у раундах:

$z[0...63] = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,$
 $1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12,$
 $5, 8, 11, 4, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2,$
 $0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9);$

4. Масив величини бітових циклічних зсувів вліво:

$s[0...63] = (7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,$
 $5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20,$
 $4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,$
 $6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21).$

Етапи виконання алгоритму:

1) Вхідні дані вирівнюються так, щоб їх розмір був рівний 448 по модулю 512.

Спочатку дописується одиничний біт (навіть якщо довжина вже дорівнює потрібній величині), потім необхідне число нульових бітів.

2) Дописуються 64-біти, що містять довжину даних до вирівнювання. Якщо

довжина перевершує $2^{64} - 1$, то дописуються молодші біти.

3) Ініціалізуються чотири 32-бітні змінні:

$A = H[0]; B = H[1]; C = H[2]; D = H[3].$

4) Кожний 512-бітний блок, представлений у вигляді 16-ти 32-розрядних значень $X[0]...X[15]$, проходить через стискаючу функцію, яка перемішує його з допоміжним блоком $(H[0], H[1], H[2], H[3])$:

цикл по j від 0 до 15

$T = (A + f(B, C, D) + x[z[j]] + y[j]) \text{ ROL } s[j]$

$(A, B, C, D) = (D, B + T, B, C)$

кінець_циклу

цикл по j від 16 до 31

$$T = (A + g(B,C,D) + x[z[j]] + y[j]) \text{ ROL } s[j]$$

$$(A,B,C,D) = (D,B+T,B,C)$$

кінець_циклу

цикл по j від 32 до 47

$$T = (A + h(B,C,D) + x[z[j]] + y[j]) \text{ ROL } s[j]$$

$$(A,B,C,D) = (D,B+T,B,C)$$

кінець_циклу

цикл по j від 48 до 63

$$T = (A + k(B,C,D) + x[z[j]] + y[j]) \text{ ROL } s[j]$$

$$(A,B,C,D) = (D,B+T,B,C)$$

кінець_циклу

$$(H[0],H[1],H[2],H[3]) = (H[0]+A,H[1]+B,H[2]+C,H[3]+D)$$

Функції, які використовуються в раундах:

- У першому раунді $f(U,V,W) = (U \text{ and } V) \text{ or } ((\text{not } U) \text{ and } W)$.
- У другому раунді $g(U,V,W) = (U \text{ and } W) \text{ or } (V \text{ and } (\text{not } W))$.
- У третьому раунді $h(U,V,W) = U \text{ xor } V \text{ xor } W$.
- У четвертому раунді $k(U,V,W) = V \text{ xor } (U \text{ or } (\text{not } W))$.

5) Підсумковий хеш = ABCD (128 бітів)

Алгоритм SHA-1

Алгоритм **SHA-1 (Secure Hash Algorithm)** – криптографічна геш-функція призначена для обчислення з повідомлення довжиною до 264 бітів значення довжиною 160 бітів. **SHA-1** розроблений у США в рамках стандарту безпечного хешування SHS (Secure Hash Standard) в 1992 р. Алгоритм SHA призначений для використання разом з алгоритмом цифрового підпису DSA.

При введенні повідомлення М довільної довжини меншої ніж 264 бітів алгоритм SHA-1 створює 160-бітове вихідне геш-значення. Потім цей дайджест повідомлення використовується як вхід алгоритму DSA, що обчислює

цифровий підпис повідомлення М. Формування цифрового підпису для дайджесту повідомлення, а не для самого повідомлення підвищує ефективність процесу підписання, оскільки дайджест повідомлення звичайно набагато коротше самого повідомлення. Такий же дайджест повідомлення повинен обчислюватися користувачем, що перевіряє отриманий підпис, при цьому як вхід в алгоритм використовується отримане повідомлення М.

Розглянемо докладніше роботу алгоритму SHA-1. Насамперед вихідне повідомлення М вирівнюють так як і при md5, щоб воно стало кратним 512 бітам, спочатку додаються одиниця, потім стільки нулів, скільки необхідно для одержання повідомлення, що на 64 біта коротше, ніж кратне 512, і нарешті додають 64-бітове подання довжини вихідного повідомлення.

Ініціалізуються п'ять 32-бітових змінних у вигляді:

$$A = 0x67452301$$

$$B = 0xEFCDAB89$$

$$C = 0x98BADCFE$$

$$D = 0x10325476$$

$$E = 0xC3D2E1F0$$

Потім починається головний цикл алгоритму. У ньому обробляється по 512 бітів повідомлення по черзі для всіх 512-бітових блоків, наявних у повідомленні. Перші п'ять змінних А, В, С, D, Е копіюються в інші змінні а, b, c, d, e:

$$a = A, b = B, c = C, d = D, e = E$$

Головний цикл містить чотири цикли по 20 операцій кожний. Кожна операція реалізує нелінійну функцію від трьох з п'яти змінних а, b, c, d, e, а потім робить зсув і додавання.

Алгоритм SHA має наступний набір нелінійних функцій:

$f_t(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$	для $t = 0 \dots 19$,
$f_t(X, Y, Z) = X \oplus Y \oplus Z$	для $t = 20 \dots 39$,
$f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	для $t = 40 \dots 59$,
$f_t(X, Y, Z) = X \oplus Y \oplus Z$	для $t = 60 \dots 79$,

В алгоритмі використовуються також чотири константи:

$$K_t = 0x5A827999 \text{ для } t = 0...19,$$

$$K_t = 0x6ED9EBA1 \text{ для } t = 20...39,$$

$$K_t = 0x8F1BBCDC \text{ для } t = 40...59,$$

$$K_t = 0xCA62C1D6 \text{ для } t = 60...79.$$

Блок повідомлення перетвориться із шістнадцяти 32-бітових слів ($M_0...M_{15}$) у вісімдесят 32-бітових слів ($W_0...W_{79}$) за допомогою наступного алгоритму:

$$W_t = M_t \text{ для } t = 0...15,$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1 \text{ для } t = 16...79,$$

де t – номер операції, W_t – t -й субблок розширеного повідомлення,

$\lll S$ – циклічне зрушення вліво на S біт.

З урахуванням уведених позначень головний цикл із вісімдесятьох операцій можна описати так:

цикл по t від 0 до 79

цикл по t від 0 до 79

$$TEMP = (a \lll 5) + f_t(b, c, d) + e + W_t + K_t$$

$$e = d$$

$$d = c$$

$$c = (b \lll 30)$$

$$b = a$$

$$a = TEMP$$

кінець_циклу

Схема виконання однієї операції показана на рис.2.

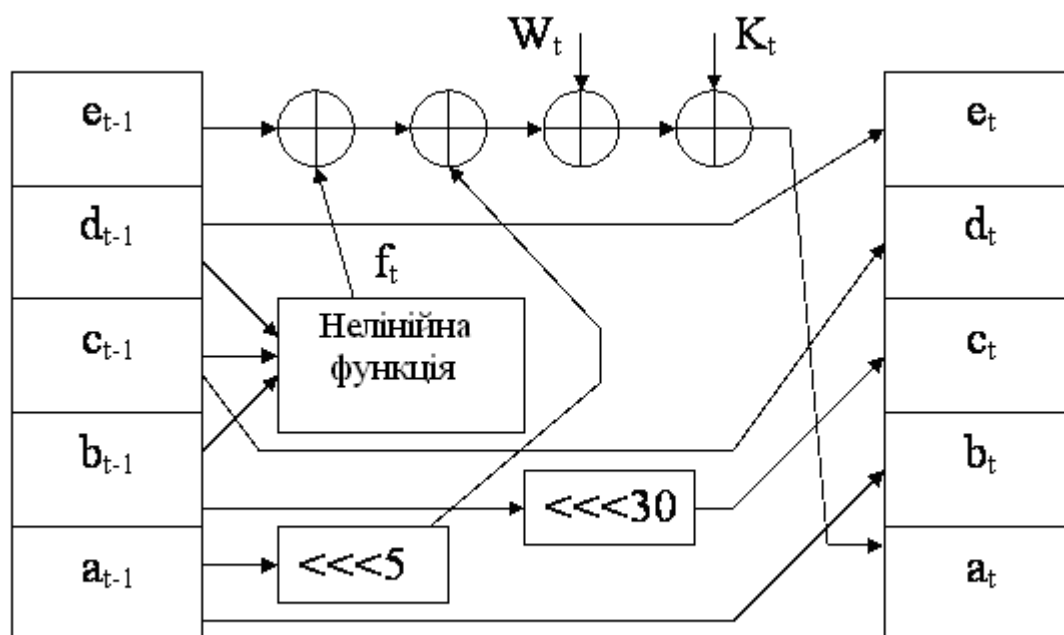


Рис.5. Схема виконання однієї операції алгоритму SHA-1

Після закінчення головного циклу значення a , b , c , d , e складаються з A , B , C , D , E відповідно, і алгоритм приступає до обробки наступного 512-бітового блоку даних. Остаточний вихід формується у вигляді конкатенації значень A , B , C , D , E .

Відмінності SHA-1 від MD5 полягають у наступному:

- SHA-1 видає 160-бітове геш-значення, тому він більш стійкий до атак повного перебору й атак "дня народження", ніж MD5, що формує 128-бітові геш-значення.
- Стискаюча функція SHA-1 складається з 80 кроків, а не з 64 як в MD5.
- Розширення вхідних даних здійснюється не простим їхнім повторенням в іншому порядку, а рекуррентною формулою.
- Ускладнено процес перемішування.

Завдання:

- Розробити програму, яка реалізує алгоритми хешування MD5 та SHA-1.
- Порівняти швидкість виконання.

Лабораторна робота №6

Тема: Автентифікація даних і електронний цифровий підпис.

Теоретичні відомості

Метою автентифікації електронних документів є їхній захист від можливих видів злочинних дій, до яких відносяться:

1. Активне перехоплення – порушник, що підключився до мережі, перехоплює документи (файли) і змінює їх;
2. Маскарад – абонент С посилає документ абоненту В від імені абонента А;
3. Ренегатство – абонент А заявляє, що не посилав повідомлення абоненту В, хоча насправді послав;
4. Підміна – абонент В змінює або формує новий документ і заявляє, що одержав його від абонента А;
5. Повтор – абонент С повторює раніше переданий документ, що абонент А надсилав абоненту В.

Для автентифікації текстів використовується електронний цифровий підпис (ЕЦП). Функціонально він аналогічний звичайному рукописному підпису й володіє його основними достоїнствами:

1. Засвідчує, що підписаний текст надходить від імені, того хто поставив підпис;
2. Не дає цій особі можливості відмовитися від зобов'язань, пов'язаних з підписаним текстом;
3. Гарантує цілісність підписаного тексту.

Загальний алгоритм створення ЕЦП.

При формуванні ЕЦП відправник насамперед обчислює геш-функцію $h(M)$ тексту M . Обчислене значення геш-функції $h(M)$ являє собою один короткий блок інформації m , що характеризує весь текст M у цілому. Потім число m шифрується секретним (закритим) ключем відправника. Одержане при цьому значення і є цифровим підписом тексту M .

При перевірці ЕЦП одержувач повідомлення знову обчислює геш-функцію $m = h(M)$ прийнятого по мережі тексту M , після чого за допомогою відкритого ключа відправника перевіряє, чи відповідає отриманий підпис обчисленому значенню m геш-функції.

Принциповим моментом у системі ЕЦП є неможливість підробки підпису без знання секретного ключа.

На відміну від шифрування під час підписання документа відкритий і закритий ключі міняються місцями. Підписання здійснюється закритим ключем, а перевірка підпису – відкритим.

Цифровий підпис на основі RSA

Першою й найбільш відомою в усьому світі конкретною системою ЕЦП стала система на основі RSA.

Алгоритм підписання документа.

Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник (автор) електронних документів обчислює два великих простих числа P і Q , потім знаходить їх добуток $N = P * Q$ і значення функції $\phi(N) = (P-1)(Q-1)$.

Далі відправник обчислює число E з умов:

$$E \leq \phi(N), \text{НСД}(E, \phi(N)) = 1$$

і число D з умов:

$$D < N, E * D \equiv 1 \pmod{\phi(N)}.$$

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписання документів.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції $h()$ у ціле число m :

$$m = h(M).$$

Потім обчислюють цифровий підпис S під електронним документом M , використовуючи геш-значення m і секретний ключ D :

$$S = m \pmod{N}.$$

Пара (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Алгоритм перевірки підпису.

Після прийому пари (M, S) одержувач обчислює геш-значення тексту M двома різними способами. Насамперед він відновлює геш-значення m' , застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E:

$$m' = S^E \pmod{N}.$$

Крім того, він знаходить результат хешування прийнятого повідомлення M за допомогою такої ж геш-функції $h()$:

$$m = h(M).$$

Якщо дотримується рівність обчислених значень m' та m :

$$S^E \pmod{N} = h(M),$$

то одержувач визнає пару (M,S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S під документом M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти модуль N на множники.

Крім того, можна суворо математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D відповідний відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" підписувача.

Верифікація відкритого ключа

Безпосереднє використання відкритих ключів вимагає додаткового їх захисту й ідентифікації для визначення зв'язку із секретним ключем. Без такого додаткового захисту зломисник може представити себе як відправником підписаних даних, так і одержувачем зашифрованих даних, замінивши значення відкритого ключа або порушивши його ідентифікацію. У цьому випадку

кожний може видати себе за англійську королеву. Все це приводить до необхідності верифікації відкритого ключа. Для цих цілей використовується електронний сертифікат.

Електронний сертифікат – цифровий документ, що зв'язує відкритий ключ із певним користувачем або додатком. Завіряє сертифікат Центр Сертифікації (ЦС). Виходячи з функцій, які виконує ЦС, він є основним компонентом всієї Інфраструктури Відкритих Ключів. За допомогою ЦС, кожний користувач може перевірити достовірність відкритого ключа. Лише сертифіковані ключі мають юридичну силу.

Завдання:

- Створити програму, яка повинна здійснювати підписання документа та перевірку підпису.

- Цифровий підпис реалізувати на основі алгоритму RSA (Лабораторні роботи №3, №5). *Розмір файлу не менш 100 кБ.*

Лабораторна робота №7

Тема: Криптоаналіз. Взлам RSA.

Теоретичні відомості

Розглянемо різні способи атак на одну з найбільш популярних схем відкритого шифрування та цифрового підпису – RSA. Розглянуті методи припускають наявність певних математичних слабостей схеми, внаслідок недосконалої програмної реалізації. Також приводяться засоби протидії цим атакам. Їх необхідно брати до уваги при реалізації схеми RSA або протоколів, заснованих на ній.

Насамперед згадаємо саму схему підпису алгоритмом RSA:

1. Вибираються p, q – великі прості числа. Обчислюється добуток $n = pq$.
2. Вибирається число e – таке, що $(e, \varphi(n)) = 1$ (тобто e та $\varphi(n)$ – взаємoprості), де $\varphi(n)$ – функція Ейлера від n .
3. З рівняння $ed = 1 \pmod{\varphi(n)}$ знаходиться число d .

Отримані числа e, n – відкритий ключ користувача, а d – секретний ключ.

Процедура шифрування: $C = E_{(e,n)}(M) = M^e \pmod{n}$, де C – зашифрований текст, M – відкритий текст, що задовольняє наступній умові: $M^{\varphi(n)} = 1 \pmod{n}$.

Процедура дешифрування: $M = D_{(d,n)}(C) = C^d \pmod{n}$.

Генерація цифрового підпису: цифровий підпис $Q = M^d \pmod{n}$.

Перевірка цифрового підпису: $Q^e \pmod{n} ? = M$.

1. Метод безключового читання RSA.

Початкові умови. Супротивникові відомий відкритий ключ (e, n) і шифротекст C .

Завдання. Знайти вихідний текст M .

Супротивник підбирає число j , для якого виконується наступне співвідношення: $C^{e^j} \pmod{n} = C$. Тобто супротивник просто здійснює j раз шифрування на відкритому ключі перехопленого шифротекста, це виглядає в такий спосіб:

$$(C^e)^e \dots)^e (\text{mod } n) = C^{e^j} (\text{mod } n).$$

Знайшовши таке j , супротивник обчислює $C^{e^{j-1}} (\text{mod } n)$, тобто $j-1$ раз повторює операцію шифрування, це значення і є відкритий текст M . Це слідує з того, що $C^{e^j} (\text{mod } n) = (C^{e^{j-1}} (\text{mod } n))^e = C$. Тобто деяке число $C^{e^{j-1}} (\text{mod } n)$ у ступені e дає шифротекст C . А що ж це, як не відкритий текст M ?

Приклад: $p = 983$, $q = 563$, $e = 49$, $M = 123456$.

$C = M^{49} (\text{mod } n) = 1603$, $C^{497} (\text{mod } n) = 85978$, $C^{498} (\text{mod } n) = 123456$, $C^{499} (\text{mod } n) = 1603$.

2. Атака на підпис RSA у схемі з нотаріусом.

Початкові умови. Є електронний нотаріус, що підписує документи, що проходять через нього. N – деякий відкритий текст, що нотаріус не бажає підписувати. Супротивникові відомий відкритий ключ (e, n) нотаріуса.

Завдання. Підписати цей текст N .

Супротивник виробляє якесь випадкове число x , взаємoprосте з N і обчислює $y = x^e (\text{mod } n)$. Потім одержує значення $M = y$ і передає його на підпис нотаріусові. Той підписує (адже це вже не текст N !) $M^d (\text{mod } n) = S$. Одержуємо, що

$$S = M^d (\text{mod } n) = y^d N^d = (x^e)^d N^d = x N^d,$$

а значить $N^d = S x^{-1} (\text{mod } n)$. Залишається просто поділити отримане S на x .

Захист. При підписі додавати деяке випадкове число в повідомлення (наприклад, час). У такий спосіб вийде викривлення числа M при підписанні.

3. Атака на підпис RSA по обраному шифротексту.

Початкові умови. Є шифротекст C . Супротивникові відомий відкритий ключ (e, n) відправника повідомлення.

Завдання. Знайти текст M

Супротивник вибирає якесь r : $r < n$, $(r, n) = 1$ і обчислює $x = r^e (\text{mod } n)$. Потім обчислює

$t = r^{-1}(\bmod n)$ і $y = x(\bmod n)$ і посилає у на підпис відправникові.

Відправник, нічого не підозрюючи, підписує текст y : $w = y^d(\bmod n)$ і відправляє w назад.

Супротивник обчислює

$$tw(\bmod n) = r^{-1}y^d(\bmod n) = (\text{оскільки } r = x^d \bmod n) = x^{-d}x^dC^d(\bmod n) = C^d = M.$$

Супротивник не може відразу послати C на підпис, тому що відправник переглядає отримані в результаті підпису повідомлення й може помітити провокацію.

Атака носить трохи гіпотетичний характер, але проте дозволяє зробити кілька важливих висновків:

- а) підписувати й шифрувати треба різними ключами, або
- б) додавати випадковий вектор при підписанні чи використовувати геш-функцію.

Завдання:

Створити програму, що реалізує один з розглянутих алгоритмів криптоаналізу RSA.

Лабораторна робота №8

Тема: Програмні закладки. Клавіатурний шпигун.

Теоретичні відомості

Програмні закладки – своєрідні програми, що використовують вірусну технологію прихованого встановлення, поширення та активізації. Однак, на відміну від вірусів, які просто знищують інформацію, програмні закладки, насамперед, призначені для її несанкціонованого одержання.

По методу впровадження в комп'ютерну систему програмні закладки можна розділити на:

- програмно-апаратні закладки, асоційовані з апаратними засобами комп'ютера (їхнє середовище перебування, як правило – BIOS);
- завантажувальні закладки, асоційовані з програмами початкового завантаження, які розташовуються в завантажувальних секторах;
- драйверні закладки, асоційовані із драйверами;
- прикладні закладки, асоційовані із прикладним програмним забезпеченням загального призначення;
- виконуючі закладки, асоційовані із виконувачими програмними модулями, що містять код цієї закладки;
- закладки-імітатори, інтерфейс яких збігається з інтерфейсом деяких службових програм, що вимагають введення конфіденційної інформації (паролів, криптографічних ключів, номерів кредитних карток);
- замасковані закладки, які маскуються під програмні засоби оптимізації роботи комп'ютера або під програми ігрового та розважального призначення.

Найпоширенішим типом закладок є клавіатурні шпигуни. Метою таких закладок є одержання інформації, що вводиться із клавіатури, (у тому числі й паролів), збереження її у зарезервованих для цього секторах, а потім пересилання накопичених даних по мережі на комп'ютер зловмисника.

Клавіатурні шпигуни діляться на імітатори, фільтри та замінювачі залежно від способу перехоплення паролів.

Імітатори працюють за наступним алгоритмом. Зловмисник впроваджує в операційну систему програмний модуль, що імітує запрошення до авторизації, щоб увійти в систему. Потім імітатор переходить у режим очікування введення ідентифікатора та пароля. Після того як потрібна інформація буде введена, програма відправляє отримані дані в місце, доступне зловмисникові й закривається. Після чого з'являється оригінал входу, у якому користувач, забивши потрібні дані, зможе потрапити в систему. Найбільш просунуті програми видають повідомлення про помилку перед своїм закриттям, щоб не викликати підозр.

Фільтри «полюють» за всіма даними, які користувач операційної системи вводить із клавіатури комп'ютера. Фільтри одержують сигнали від клавіатури, які далі аналізуються на предмет виявлення даних про різні паролі. Найпростіший фільтр збирає в домовленому місці всі введені із клавіатури дані, більш витончені здатні відфільтровувати тільки ті дані, які необхідні зловмисникові.

Замінювачі повністю або частково підмінюють собою програмні модулі операційної системи, відповідальні за автентифікацію користувачів. Перед виконанням замінювачам необхідно:

- 1) подібно комп'ютерному вірусу впровадитися в один або кілька системних файлів;
- 2) використовувати інтерфейсні зв'язки між програмними модулями системи автентифікації для вбудовування себе в ланцюжок обробки введеного користувачем пароля.

Крім того, багато клавіатурних шпигунів відслідковують список запущених додатків, уміють робити "знімки" екрана за заданим розкладом, шпигувати за вмістом буфера обміну й вирішувати ряд завдань, націлених на приховане спостереження за користувачем. Записувана інформація зберігається на диску й більшість сучасних клавіатурних шпигунів можуть формувати різні звіти, можуть передавати їх по електронній пошті або http/ftp протоколу. Крім того, ряд сучасних клавіатурних шпигунів користуються RootKit технологіями

для маскуванню слідів своєї присутності в системі.

Перед описанням основних принципів роботи клавіатурного шпигуна необхідно розглянути модель апаратного введення системи Windows.

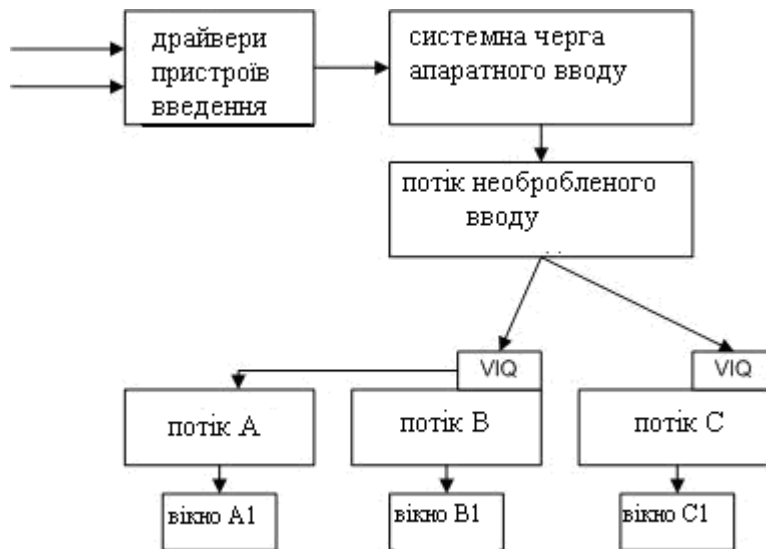


Рисунок 1 – Модель апаратного введення системи Windows

При виникненні будь-яких подій введення (натисканні клавіш, переміщенні миші) події обробляються відповідним драйвером і поміщаються в системну чергу апаратного введення. У системі є особливий потік неопрацьованого введення, який називається RIT (Raw Input Thread), що витягає події із системної черги та перетворює у повідомлення. Отримані повідомлення поміщаються в кінець черги віртуального введення одного з потоків (віртуальна черга потоку називається VIQ – Virtualized Input Queue). При цьому RIT сам з'ясовує, у чергу якого конкретно потоку необхідно помістити подію. Для подій миші потік визначається пошуком вікна, над яким розташований курсор миші. Клавіатурні події відправляються тільки одному потоку – так званому активному потоку (тобто потоку, якому належить вікно, з яким працює користувач). Насправді це не зовсім так – зокрема, на рисунку показаний потік А, що не має черги віртуального введення. У цьому випадку виходить, що потоки А і В спільно використовують одну чергу віртуального

введення. Це досягається за допомогою виклику API функції `AttachThreadInput`, що дозволяє одному потоку підключитися до черги віртуального введення іншого потоку.

Слід зазначити, що потік неопрацьованого введення відповідає за обробку спеціальних сполучень клавіш, зокрема `Alt+Tab` і `Ctrl+Alt+Del`.

Спостереження за клавіатурним вводом за допомогою пасток

Дана методика є класичною для клавіатурних шпигунів. Суть методу складається в застосуванні механізму пасток (hook) операційної системи. Пастки дозволяють спостерігати за повідомленнями, які обробляються вікнами інших програм. Установка та видалення пасток виконується за допомогою добре документованих функцій API бібліотеки `user32.dll` (функція `SetWindowsHookEx` дозволяє встановити пастку, `UnhookWindowsHookEx` – зняти її). При установці пастки вказується тип повідомлень, для яких повинен викликатися оброблювач пастки. Зокрема, є два спеціальних типи пастки `WH_KEYBOARD` і `WH_MOUSE` – для реєстрації подій клавіатури й миші відповідно. Пастка може бути встановлена для заданого потоку й для всіх потоків системи. Пастка для всіх потоків системи дуже зручна для побудови клавіатурного шпигуна.

Код оброблювача подій пастки повинен бути розташований в DLL. Ця вимога пов'язана з тим, що DLL з оброблювачем пастки проектується системою в адресний простір всіх GUI процесів. Цікавою особливістю є те, що проектування DLL відбувається не в момент установки пастки, а при одержанні GUI процесом першого повідомлення, що задовольняє параметрам пастки.

Методика пасток досить проста й ефективна, але в ній є ряд недоліків. Першим недоліком можна вважати те, що DLL з пасткою проектується в адресний простір всіх GUI процесів, що може застосовуватися для виявлення клавіатурного шпигуна. Крім того, реєстрація подій клавіатури можлива тільки для GUI додатків.

Спостереження за клавіатурним вводом за допомогою опитування клавіатури

Дана методика заснована на періодичному опитуванні стану клавіатури. Для опитування стану клавіш у системі передбачена спеціальна функція `GetKeyboardState`, що повертає масив з 255 байт, у якому кожний байт містить стан певної клавіші на клавіатурі. Даний метод уже не вимагає впровадження DLL в GUI процеси й у результаті шпигун менш помітний.

Однак зміна статусу клавіш відбувається в момент зчитування потоком клавіатурних повідомлень із його черги, і в результаті подібна методика працює тільки для спостереження за GUI додатками. Від цього недоліку звільнена функція `GetAsyncKeyState`, що повертає стан клавіші на момент виклику функції.

Недоліком клавіатурних шпигунів такого типу є необхідність періодичного опитування стану клавіатури з досить високою швидкістю, не менш 10-20 опитувань у секунду.

Спостереження за клавіатурним вводом за допомогою перехоплення API функцій

Дана методика не одержала широкого поширення, але проте вона може з успіхом застосовуватися для побудови клавіатурних шпигунів. Найпростішим способом може бути перехоплення функцій `GetMessage`, `PeekMessage` і `TranslateMessage` бібліотеки `User32`, що дозволить вести моніторинг всіх повідомлень, прийнятих GUI додатками.

Клавіатурний шпигун на базі драйверу

Даний метод ще більш ефективний, ніж описані вище методи. Можливі як мінімум два варіанти реалізації цього методу – написання та установка в систему свого драйвера клавіатури замість штатного або встановлення драйвера – фільтра.

При створенні свого драйверу треба перехоплювати переривання від

клавіатури (IRQ 1) та напряду звертатися до портів вводу-виводу (60h, 64h).

Приклади драйверів-фільтрів: драйвер-фільтр драйвера класу клавіатури Kbdclass, драйвер-фільтр функціонального драйвера i8042prt. *Драйвер-фільтр драйвера класу клавіатури Kbdclass* перехоплює запити до клавіатури за допомогою встановлення фільтра зверху пристрою "\Device\KeyboardClass0", створеного драйвером Kbdclass. Фільтруються тільки запити типу IRP_MJ_READ, оскільки саме вони дозволяють одержати коди натиснутих і відпущених клавіш. *Драйвер-фільтр функціонального драйвера i8042prt* перехоплює запити до клавіатури за допомогою встановлення фільтра зверху безіменного пристрою, що створюється драйвером i8042prt під пристроєм Device\KeyboardClass0. Драйвер i8042prt представляє собою програмний інтерфейс для додавання додаткової функції обробки переривання IRQ1 (IsrRoutine), у якій можна зробити аналіз отриманих від клавіатури даних.

Апаратні клавіатурні шпигуни

До них належать:

- Установка пристрою спостереження в розрив кабелю клавіатури (наприклад, пристрій може бути виконаний у вигляді перехідника PS/2);
- Вбудовування пристрою спостереження в клавіатуру;
- Зчитування даних шляхом реєстрації побічних електромагнітних випромінювань і наведень;
- Візуальне спостереження за клавіатурою.

Апаратні клавіатурні шпигуни зустрічаються набагато рідше, ніж програмні. Однак при перевірці особливо відповідальних комп'ютерів (наприклад, тих, що використовуються для здійснення банківських операцій) про можливість апаратного спостереження за клавіатурним введенням не слід забувати.

Методики пошуку клавіатурних шпигунів

Пошук по сигнатурах. Даний метод не відрізняється від типових методик пошуку вірусів. Сигнатурний пошук дозволяє однозначно ідентифікувати клавіатурні шпигуни, при правильному виборі сигнатур ймовірність помилки практично дорівнює нулю. Однак сигнатурний сканер зможе виявляти заздалегідь відомі й описані в його базі дані об'єкти;

Евристичні алгоритми. Як очевидно з назви, це методики пошуку клавіатурного шпигуна по його характерних рисах. Евристичний пошук носить ймовірнісний характер. Як показала практика, цей метод найбільш ефективний для пошуку клавіатурних шпигунів найпоширенішого типу – заснованих на пастках. Однак подібні методики дають багато помилкових спрацьовувань. Існують сотні безпечних програм, що не є клавіатурними шпигунами, але використовують пастки для спостереження за клавіатурним вводом і мишею. Найпоширеніші приклади – програми Punto Switcher, словник Lingvo, програмне забезпечення від мультимедійних клавіатур і мишей;

Моніторинг API функцій. Дана методика заснована на перехопленні ряду функцій, що застосовуються клавіатурним шпигуном – зокрема, функцій SetWindowsHookEx, UnhookWindowsHookEx, GetAsyncKeyState, GetKeyboardState. Виклик даних функцій будь-яким додатком дозволяє вчасно підняти тривогу, однак проблеми численних помилкових спрацьовувань будуть аналогічні методу 2;

Відстеження драйверів, процесів і сервісів, що використовуються системою. Це універсальна методика, що застосовується не тільки проти клавіатурних шпигунів а й проти вірусів. Цей метод застосовують Kaspersky Inspector або Adinf, які відслідковують появу в системі нових файлів.

Завдання:

Створити клавіатурний шпигун, що при кожному запуску створює файл у який записує дату, ім'я користувача, програму, до якої йде звертання і текст набраний на клавіатурі. А також створити програму, що знаходить та знищує цю програмну закладку. Реалізувати будь-яким з вище перерахованих методів.

Лабораторна робота №9

Тема: Стеганографія

Теоретичні відомості

Стеганографія – це наука про приховану передачу інформації шляхом збереження в таємниці самого факту передачі. На відміну від криптографії, що приховує вміст секретного повідомлення, стеганографія приховує саме його існування.

Наприкінці 90-х років виділилося кілька напрямків стеганографії: класична, комп'ютерна та цифрова.

Кілька прикладів класичної стеганографії: використання «невидимих» чорнил; запис на бічній стороні колоди карт, розташованих у домовленому порядку; «жаргонні шифри», де слова мають інше обумовлене значення; трафарети, які будучи покладеними на текст, залишають видимими тільки значущі букви; вузлики на нитках і т.д.

Комп'ютерна стеганографія – напрямок класичної стеганографії, заснований на особливостях комп'ютерної платформи. Приклади: стеганографічна файлова система StegFS для Linux, приховання даних у невикористовуваних областях форматів файлів, підміна символів у назвах файлів, текстова стеганографія й т.д.

Цифрова стеганографія – напрямок класичної стеганографії, заснований на прихованні або вбудовуванні додаткової інформації в цифрові об'єкти. Але, як правило, дані об'єкти є мультимедіа-об'єктами (зображення, відео, аудіо, текстури 3D-об'єктів). При цьому файли звичайно спотворюються, але такі спотворення перебувають нижче порога чутливості середньостатистичної людини і не приводять до помітних змін цих об'єктів. Крім того в оцифрованих об'єктах, що споконвічно мають аналогову природу, і так завжди є присутнім шум.

Застосування цифрової стеганографії

Цифрова стеганографія як наука народилася буквально в останні роки. Вона містить у собі наступні напрямки:

- 1) вбудовування інформації з метою її прихованої передачі;
- 2) вбудовування цифрових водяних знаків (ЦВЗ) (watermarking);
- 3) вбудовування ідентифікаційних номерів (fingerprinting);
- 4) вбудовування заголовків (captioning).

Найбільш поширений легальний напрямок стеганографії – вбудовування цифрових водяних знаків, що є основою для систем захисту авторських прав. Методи цього напрямку спрямовані на вбудовування прихованих маркерів, стійких до різних спотворень контейнера (архівації, атак зловмисників тощо). Вбудовуватися може як текст, так і малюнок невеликого розміру.

Алгоритми

По способу вбудовування інформації стегоалгоритми можна розділити на лінійні (адитивні), нелінійні та інші.

Найбільш відомими представниками *прямих методів* є:

LSB-методи (Least Significant Bits);

Методи модифікації палітри.

Нелінійні:

дискретне косинус-перетворення (ДКП);

вейвлет-перетворення;

дискретне перетворення Фур'є;

перетворення Карунена-Лоєва;

сингулярне розкладання.

До інших відноситься наприклад метод на основі фрактального перетворення.

Метод LSB

LSB (Least Significant Bit, найменший значущий біт) – суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображенні, аудіо– чи відеозаписі) на біти приховуваного повідомлення. Різниця між порожнім і

заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Методи LSB є нестійкими до всіх видів атак і можуть бути використані тільки при відсутності шуму в каналі передачі даних. Також наявність прихованого повідомлення легко виявити. Виявлення такого стегоповідомлення здійснюється по аномальних характеристиках розподілу значень діапазону молодших бітів цифрового сигналу (рис. 1).

Всі методи LSB являються як правило адитивними.

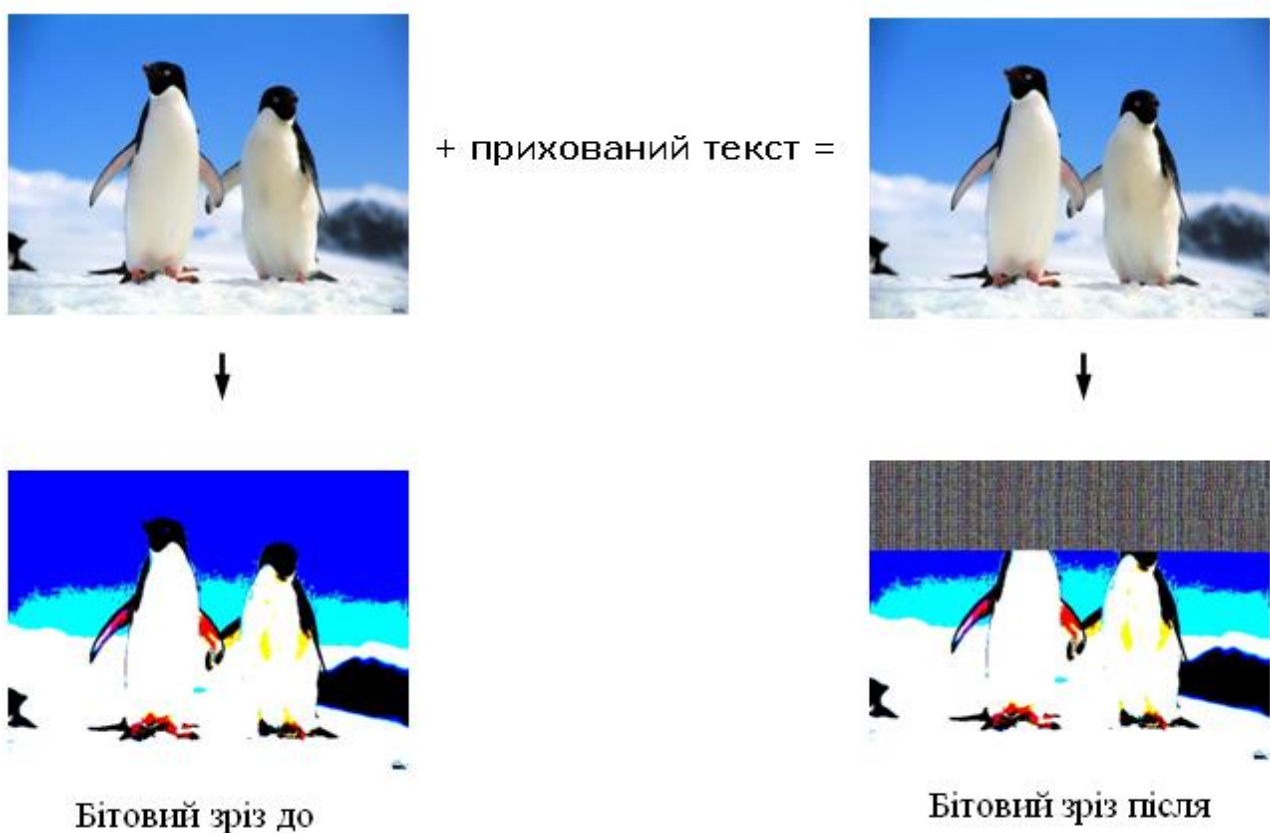


Рисунок 1 – Вигляд бітового зрізу файлу до та після вдубовування текстового повідомлення методом LSB.

Найлегше даний метод застосовувати для *.bmp* файлів.

Формат *bmp* (від слів BitMap – бітова карта, бітовий масив) – один з растрових форматів операційної системи Windows, який запам'ятовує одно–і

багатокольорові (RGB) зображення у формі Pixel.

Структура Bmp-файлу

Ім'я	Довжина	Зсув	Опис
Заголовок файлу растрової графіки (BitmapFileHeader) (14 байт)			
Type	2	0	Сигнатура (Код 4D42h – Букви 'BM')
Size	4	2	Розмір файлу
Reserved 1	2	6	Зарезервовано
Reserved 2	2	8	Зарезервовано
OffsetBits	4	10	Місцезнаходження даних растрового масива
Інформаційний заголовок растрового масива (BitmapInfoHeader) (40 байт)			
Size	4	14	Довжина цього заголовка
Width	4	18	Ширина зображення (в пікселях)
Height	4	22	Висота зображення (в пікселях)
Planes	2	26	Число площин
BitCount	2	28	Глибина кольору, біт/піксель
Compression	4	30	Тип компресії (0 – нестиснуте зображення)
SizeImage	4	34	Розмір зображення, байт
XpelsPerMeter	4	38	Горизонтальне розрішення, піксель/метр
YpelsPerMeter	4	42	Вертикальне розрішення, піксель/метр
ColorsUsed	4	46	Число кольорів, що використовуються
ColorsImportant	4	50	Число основних кольорів
Таблиця кольорів (ColorTable) (довжина змінюється від 8 до 1024 байт)			
ColorTable	4*N	x	палітра (256 елементів по 4 байтив для 256-кольорового зображення)
Дані зображення (Bitmap Array)			
Image	Size	x	Зображення, записане по рядках з ліва на право і знизу вгору

ВАЖЛИВО!

1. Зображення зберігається по-рядково знизу-догори. Для збереження кожного рядка виділяється кратна 4 кількість байт. У незначущих байтах зберігається сміття. Старшому біту або тетрадї відповідає самий лівий піксель.

2. Не всі файли BMP мають таку структуру. Наприклад, файли BMP із глибиною 16 і 24 біт/піксель не мають таблиць кольорів; у цих файлах значення

пікселів растрового масиву безпосередньо характеризують значення кольорів RGB.

При зберіганні зображення True Color кожному пікселю відповідають три послідовні байти, що зберігають складові кольори B, G, R; (не R, G, B).

Елемент палітри являє собою чотирьохбайтовий запис (структуру). У цій структурі зберігаються складові R-червоного, G-зеленого та B-синього кольорів. Один байт зарезервований. Палітра може й не використовуватися, наприклад в True Color.

Структура елемента палітри:

```
typedef struct tagRGBQUAD
{
    char  rgbBlue;
    char  rgbGreen;
    char  rgbRed;
    char  rgbReserved;
} RGBQUAD;
```

У полі Тип стиснення може стояти:

- 0 – стиснення не використовується
- 1 – RLE8 стиснення (для 256-ти кольорового зображення)
- 2 – RLE4 стиснення (для 16-ти кольорового зображення)

У полі Глибина кольору:

- 1 – чорно-біле зображення
- 4 – 16-ти кольорове
- 8 – 256-ти кольорове
- 24 – True Color

```
typedef unsigned long DWORD; // Подвійне слово – 32 біти (розряди)
typedef unsigned int WORD;   // Слово – 16 бітів (розрядів)
```



```

typedef signed long LONG;
typedef unsigned int UINT;
// Заголовок файлу
typedef struct tagBITMAPFILEHEADER
{
    UINT bfType;          // 'BM' = 4D42h
    DWORD bfSize;
    UINT bfReserved1;
    UINT bfReserved2;
    DWORD bfOffBits;      // Зсув до растра
} BITMAPFILEHEADER;
// Заголовок Bitmap
typedef struct tagBITMAPINFOHEADER
{
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER;

```

Завдання:

Розробити програму, яка реалізує метод стеганографії LSB для зображень формату .bmp. Програма повинна вбудовувати та вилучати приховане повідомлення.

Список використаної літератури

1. Хорошко В.А. Методы и средства защиты информации / Хорошко В.А., Чекатков А.А. / Под ред. Ю.С. Ковтанюка. – К.: Юниор, 2003. – 504 с.
2. Термінологічний довідник з питань технічного захисту інформації / Коженевський С.Р., Кузнецов Г.В., Хорошко В.О., Чирков Д.В. / За ред. проф. В.О. Хорошка. – К.: ДУІКТ, 2007. – 365 с.
3. Макс Ронге. Разведка и контрразведка / М. Ронге /. – К.: СИНТО, 1993. – 239 с.
4. Мухачев В.А. Методы практической криптографии / Мухачев В.А., Хорошко В.А./ . – К.: ПолиграфКонсалтинг, 2005. – 214 с.
5. Мицан И.Б. Стеганографические методы сокрытия информации / Мицан И.Б. // Специальная техника и вооружение. Научно-методическое издание. – К., № 1 – 5, 2001. – С. 28 – 32.
6. Хорошко В.О. Основы комп'ютерної стеганографії. Навчальний посібник / В.О. Хорошко, О.Д. Азаров, М.Є. Шелест, Ю.Є. Яремчик /. – Вінниця: ВДТУ, 2003. – 143 с.
7. Конахович Г.Ф. Компьютерная стеганография. Теория и практика / Конахович Г.Ф., Пузыренко А.Ю. /. – К.: «МК-Пресс», 2006. – 288 с.
8. Безопасность информационных технологий. Методология создания систем защиты/ В.В. Домарев. – К.: ООО "ТИД "ДС", 2001. – 688 с.
9. Энциклопедия промышленного шпионажа/ Под общ. ред. Е.В. Куренкова – С.Петербург: ООО "Изд-во Полигон", 1999. – 512 с.
10. Хорев А;А. Способы и средства защиты информации. – М.: МО РФ, 2000. -316 с.
11. А.Ю.Щербаков. Введение в теорию и практику компьютерной безопасности. -М.: издатель Молгачева С.В., 2001.
12. Бармен, Скотт. Разработка правил информационной безопасности.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002.
13. С.Л.Емельянов Основы информационной безопасности.– Одесса:

Юридична література, 2003.-198с.

14.Про державну таємницю. Закон України №3855-ХІІ від 21.01.1994 р. (в редакції Закону № 1079-14 від 21.09.99).

15.Про затвердження зводу відомостей, що становлять державну таємницю. Наказ Голови Служби безпеки України від 12.08.2005 р. № 440.

16.НД ТЗІ 1.1 – 002 – 99. Общие положения по защите информации в компьютерных системах от несанкционированного доступа. Нормативний документ ДСТЗИ СБ України. Киев, 1999.

17.Про інформацію. Закон України № 2657-ХП від 02.10.92 р.

18.Концепція технічного захисту інформації в Україні. Постанова КМУ №1126 8.10.97.

19.Положення про технічний захист інформації в Україні. Указ Президента України №1229/99 від 27.09.99.

20.Тимчасові рекомендації з технічного захисту інформації від витоку каналами побічних електромагнітних випромінювань і наводок. (ТР ТЗІ-ПЕМВН-95). Затверджені наказом ДСТЗИ від 09.06.95р. № 25.

21.НД ТЗІ 1.4-001-2000. Типове положення про службу захисту інформації в автоматизованій системі.

22.НД ТЗІ 2.5-005-99. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу.

23.НД ТЗІ 2.1-001-2001. Створення комплексів технічного захисту інформації. Атестація комплексів. Основні положення.

24.НД ТЗІ 3.7-001-99. Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі (Зі зміною № 1, затвердженою наказом ДСТСЗІСБУ 18.06.02 № 37).

25.НД ТЗІ 2.5-010-2003. Вимоги до захисту інформації \№ЕВ-сторінки від несанкціонованого доступу.

26.13.ГОСТ Р 51275-99. Защита информации. Объект информатизации.

Факторы, воздействующие на информацию. Общие положения.

27. <http://www.intuit.ru/department/security/secbasics/>

28. Галатенко В.А. Основы информационной безопасности Интернет-университет информационных технологий – ИНТУИТ.ру, 2008

29. Лапониная О.Р. Основы сетевой безопасности: криптографические алгоритмы и протоколы взаимодействия Интернет-университет информационных технологий – ИНТУИТ.ру, 2005

30. Галатенко В.А. Стандарты информационной безопасности Интернет-университет информационных технологий – ИНТУИТ.ру, 2005

31. Э. Мэйволд Безопасность сетей Эком, 2006

32. Хаулет Т. Защитные средства с открытыми исходными текстами БИНОМ. Лаборатория знаний, Интернет-университет информационных технологий – ИНТУИТ.ру, 2007

33. Department of Defense Trusted Computer System Evaluation Criteria DoD 5200.28-STD, 1993.

34. Information Technology Security Evaluation Criteria (ITSEC). Harmonized Criteria of France – Germany – the Netherlands – the United Kingdom Department of Trade and Industry, London, 1991.

35. Security Architecture for Open Systems Interconnection for CCITT Applications. Recommendation X.800 CCITT, Geneva, 1991.

36. Site Security Handbook. Holbrook P., Reynolds J., Request for Comments: 1244, 1991.

37. James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, Edward Roback Report on the Development of the Advanced Encryption Standard (AES) Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Technology Administration U.S. Department of Commerce. 2000г. 116с.

38. Государственный Стандарт Российской Федерации Информационная технология. Криптографическая защита информации. Процедуры выработки и

проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма 1994г.

39.Государственный Стандарт Российской Федерации Информационная технология. Криптографическая защита информации. Функция хэширования 1994г.

40.RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile 2002г. 129с.

41.RFC 3281 An Internet Attribute Certificate Profile for Authorization 2002г. 40с.

42.RFC 2510 Internet X.509 Public Key Infrastructure Certificate Management Protocols 1999г. 72с.

43.RFC 2511 Internet X.509 Certificate Request Message Format 1999г. 25с.

44.RFC 3369 Cryptographic Message Syntax 2002г. 60с.

45.RFC 2560 X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP 1999г. 23с.

46.RFC 2797 Certificate Management Messages over CMS 2000г. 47с.

47.RFC 3379 Delegated Path Validation and Delegated Path Discovery Protocol Requirements 2002г. 15с.

48.RFC 2633 S/MIME Version 3 Message Specification 1999г. 32с.

49.RFC 2632 S/MIME Version 3 Certificate Handling 1999г. 13с.

50.Security Architecture for the Internet Protocol RFC 2401 1998г. 66с.

51.Internet Security Association and Key Management Protocol (ISAKMP) RFC 2408 1998г. 86с.

52.The Internet Key Exchange (IKE) RFC 2409 1998г. 41с.

53.RFC 2412 The OAKLEY Key Determination Protocol 1998г. 55с.

Інформаційні ресурси

Бібліотеки, Інтернет, електронні книги.