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

Теорія захисту інформації

Методичні рекомендації до виконання контрольних робіт для студентів заочної форми навчання галузі 12 Інформаційні технології

ЗАТВЕРДЖЕНО
на засіданні кафедри кібербезпеки та програмного забезпечення, протокол № 1 від 15.08.2022

Кропивницький
2023

Теорія захисту інформації: Методичні рекомендації до виконання контрольних робіт для студентів заочної форми навчання галузі 12 Інформаційні технології. – Кропивницький: ЦНТУ – 2023. – 61 с./М-во освіти і науки України, Центральноукр. нац. техн. ун-т; /уклад. Смірнова Т.В., Буравченко К.О., Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А. / – Кропивницький: ЦНТУ – 2023. – 61 с.

Укладачі: Смірнова Т.В., Буравченко К.О., Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А.

Рецензенти: Коваленко О.В., докт. техн. наук, доцент;
Улічев О.С., канд. техн. наук.

ЗМІСТ

ВСТУП……..……4

Контрольна робота №1. Закони України стосовно захисту інформації ….10

Контрольна робота №2. Симетричні алгоритми шифрування. DES ……...11

Контрольна робота №3. Асиметричні алгоритми шифрування. RSA …....20

Контрольна робота №4. Генератори псевдовипадкових чисел. Геш-функції ……..……23

Контрольна робота №5. Автентифікація користувача і електронний цифровий підпис.……..…….36

Контрольна робота №6. Криптоаналіз. Взлам……...….40

Контрольна робота №7. Програмні закладки. Клавіатурний шпигун. Стеганографія.……..…….43

ВСТУП

Метою освітньої компоненти «Теорія захисту інформації» є формування у здобувачів вищої освіти ґрунтовних теоретичних знань, практичних умінь та навичок, необхідних для застосування в професійній діяльності у сфері захисту інформації.

Основними **завданнями** вивчення дисципліни є формування наступних **компетенцій магістра з комп'ютерних наук:**

– СК05. Здатність розробляти, описувати, аналізувати та оптимізувати архітектурні рішення інформаційних та комп'ютерних систем різного призначення.

– СК07. Здатність розробляти програмне забезпечення відповідно до сформульованих вимог з урахуванням наявних ресурсів та обмежень.

У результаті вивчення дисципліни студент повинен забезпечити наступні **програмні результати навчання**:

– РН9. Розробляти алгоритмічне та програмне забезпечення для аналізу даних (включно з великими).

– РН10. Проектувати архітектурні рішення інформаційних та комп'ютерних систем різного призначення.

Основними **завданнями** вивчення дисципліни є формування наступних **компетенцій магістра з комп'ютерної інженерії:**

– СК1. Здатність до визначення технічних характеристик, конструктивних особливостей, застосування і експлуатації програмних, програмно-технічних засобів, комп'ютерних систем та мереж різного призначення.

– СК6. Здатність використовувати та впроваджувати нові технології, включаючи технології розумних, мобільних, зелених і безпечних обчислень, брати участь в модернізації та реконструкції комп'ютерних систем та мереж, різноманітних вбудованих і розподілених додатків, зокрема з метою підвищення їх ефективності.

– СК10. Здатність ідентифікувати, класифікувати та описувати роботу програмно-технічних засобів, комп’ютерних систем, мереж та їхніх компонентів.

У результаті вивчення дисципліни студент повинен забезпечити наступні **програмні результати навчання:**

– РН2. Знаходити необхідні дані, аналізувати та оцінювати їх.

– РН4. Застосовувати спеціалізовані концептуальні знання, що включають сучасні наукові здобутки у сфері комп’ютерної інженерії, необхідні для професійної діяльності, оригінального мислення та проведення досліджень, критичного осмислення проблем інформаційних технологій та на межі галузей знань.

– РН8. Застосовувати знання технічних характеристик, конструктивних особливостей, призначення і правил експлуатації програмно-технічних засобів комп’ютерних систем та мереж для вирішення складних задач комп’ютерної інженерії та дотичних проблем.

– РН9. Розробляти програмне забезпечення для вбудованих і розподілених застосувань, мобільних і гібридних систем.

– РН11. Приймати ефективні рішення з питань розроблення, впровадження та експлуатації комп’ютерних систем і мереж, аналізувати альтернативи, оцінювати ризики та імовірні наслідки рішень.

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 мережеву безпеку Інтернет. Віруси й антивіруси; Адміністративний рівень

інформаційної безпеки; Безпеку операційних систем; Стеганографію; Біометричні методи захисту інформації

– **вміти:** Програмно реалізовувати наступні проекти: База даних щодо законів України стосовно захисту інформації; Симетричні алгоритми шифрування; Асиметричні алгоритми шифрування; Генератори псевдовипадкових чисел. Геш-функції; Автентифікація користувача і електронний цифровий підпис; Криптоаналіз; Програмні закладки. Клавіатурний шпигун. Стеганографія.

Структурно логічна схема підготовки магістра

Враховуючи послідовність накопичення знань та інформації, бажано отримання на першому (бакалаврському) рівні вищої освіти знань з наступних дисциплін: «Вища математика», «Алгоритми та методи обчислень», «Базові методології та технології програмування», «Бази даних», «Інженерія програмного забезпечення», «Системне програмне забезпечення», «Комп’ютерні мережі».

Для опанування матеріалу дисципліни «Теорія 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, заліку (денна та заочна).

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

Контроль знань

Критерії оцінки іспиту:

оцінку «відмінно» (90-100 балів, А) – заслуговує студент, який:

– всебічно, систематично і глибоко володіє навчально-програмовим матеріалом;

– вміє самостійно виконувати завдання, передбачені програмою, використовує набуті знання і вміння у нестандартних ситуаціях;

– засвоїв основну і ознайомлений з додатковою літературою, яка рекомендована програмою;

– засвоїв взаємозв'язок основних понять дисципліни та усвідомлює їх значення для професії, яку він набуває;

– вільно висловлює власні думки, самостійно оцінює різноманітні життєві явища і факти, виявляючи особистісну позицію;

– самостійно визначає окремі цілі власної навчальної діяльності, виявив творчі здібності і використовує їх при вивченні навчально-програмового матеріалу, проявив нахил до наукової роботи.

оцінку « добре» (82-89 балів, В) – заслуговує студент, який:

– повністю опанував і вільно (самостійно) володіє навчально-програмовим матеріалом, в тому числі застосовує його на практиці, має системні знання достатньому обсязі відповідно до навчально-програмового матеріалу, аргументовано використовує їх у різних ситуаціях;

– має здатність до самостійного пошуку інформації, а також до аналізу, постановки і розв'язування проблем професійного спрямування;

– під час відповіді допустив деякі неточності, які самостійно виправляє, добирає переконливі аргументи на підтвердження вивченого матеріалу;

оцінку «добре» (74-81 бал, C) заслуговує студент, який:

– в загальному роботу виконав, але відповідає на екзамені з певною кількістю помилок;

– вміє порівнювати, узагальнювати, систематизувати інформацію під керівництвом викладача, в цілому самостійно застосовувати на практиці, контролювати власну діяльність;

– опанував навчально-програмовий матеріал, успішно виконав завдання, передбачені програмою, засвоїв основну літературу, яка рекомендована програмою;

оцінку «задовільно» (64-73 бали, D) – заслуговує студент, який:

– знає основний навчально-програмовий матеріал в обсязі, необхідному для подальшого навчання і використання його у майбутній професії;

– виконує завдання, але при рішенні допускає значну кількість помилок;

– ознайомлений з основною літературою, яка рекомендована програмою;

– допускає на заняттях чи екзамені помилки при виконанні завдань, але під керівництвом викладача знаходить шляхи їх усунення.

оцінку «задовільно» (60-63 бали, E) – заслуговує студент, який:

– володіє основним навчально-програмовим матеріалом в обсязі, необхідному для подальшого навчання і використання його у майбутній

професії, а виконання завдань задовольняє мінімальні критерії. Знання мають репродуктивний характер.

оцінка «незадовільно» (35-59 балів, FX) – виставляється студенту, який:

– виявив суттєві прогалини в знаннях основного програмового матеріалу, допустив принципові помилки у виконанні передбачених програмою завдань.

оцінку «незадовільно» (35 балів, F) – виставляється студенту, який:

– володіє навчальним матеріалом тільки на рівні елементарного розпізнавання і відтворення окремих фактів або не володіє зовсім;

– допускає грубі помилки при виконанні завдань, передбачених програмою;

– не може продовжувати навчання і не готовий до професійної діяльності після закінчення університету без повторного вивчення даної дисципліни.

При виставленні оцінки враховуються результати навчальної роботи студента протягом семестру

Критерії оцінки заліку:

– «зараховано» – студент має стійкі знання про основні поняття дисципліни, може сформулювати взаємозв'язки між поняттями.

– «незараховано» – студент має значні пропуски в знаннях, не може сформулювати взаємозв'язку між поняттями, що вивчаються в курсі, не має уявлення про більшість основних понять дисципліни, що вивчається.

Шкала оцінювання: національна та ЄКТС

<table>
<tr><td rowspan="2">Сума балів за всі види навчальної діяльності</td><td rowspan="2">Оцінка ЄКТС</td><td colspan="2">Оцінка за національною шкалою</td></tr>
<tr><td>для екзамену, курсового проекту (роботи), практики</td><td>для заліку</td></tr>
<tr><td>90-100</td><td>A</td><td>відмінно</td><td rowspan="5">зараховано</td></tr>
<tr><td>82-89</td><td>B</td><td rowspan="2">добре</td></tr>
<tr><td>74-81</td><td>C</td></tr>
<tr><td>64-73</td><td>D</td><td rowspan="2">задовільно</td></tr>
<tr><td>60-63</td><td>E</td></tr>
<tr><td>35-59</td><td>FX</td><td>незадовільно з можливістю повторного складання</td><td>не зараховано з можливістю повторного складання</td></tr>
<tr><td>1-34</td><td>F</td><td>незадовільно з обов’язковим повторним вивченням дисципліни</td><td>не зараховано з обов’язковим повторним вивченням дисципліни</td></tr>
</table>

Контрольна робота № 1

Тема: Закони України стосовно захисту інформації

Посилання на закони та стандарти дивіться в лекціях № 1-2. Закони можна переглянути на сайті Верховної Ради України (http://portal.rada.gov.ua/) й Державної служби спеціального зв'язку та захисту інформації України (http:// https://cip.gov.ua/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.

Контрольна робота № 2

Тема: Симетричні алгоритми шифрування. DES.

Розглянемо загальну схему симетричної, або традиційної, криптографії.

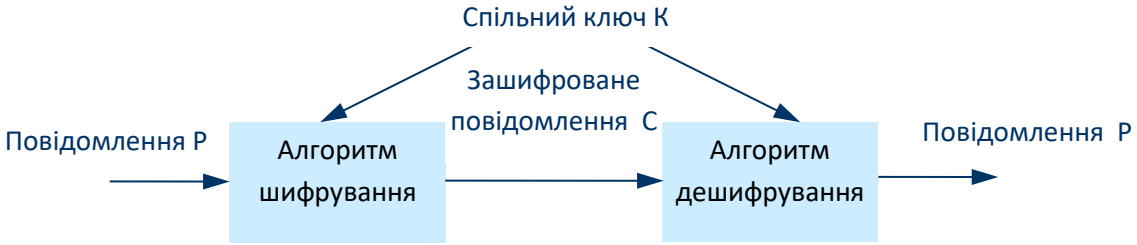

Рис. 1. Загальна схема симетричного шифрування

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 перетворять 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 Technolody 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.

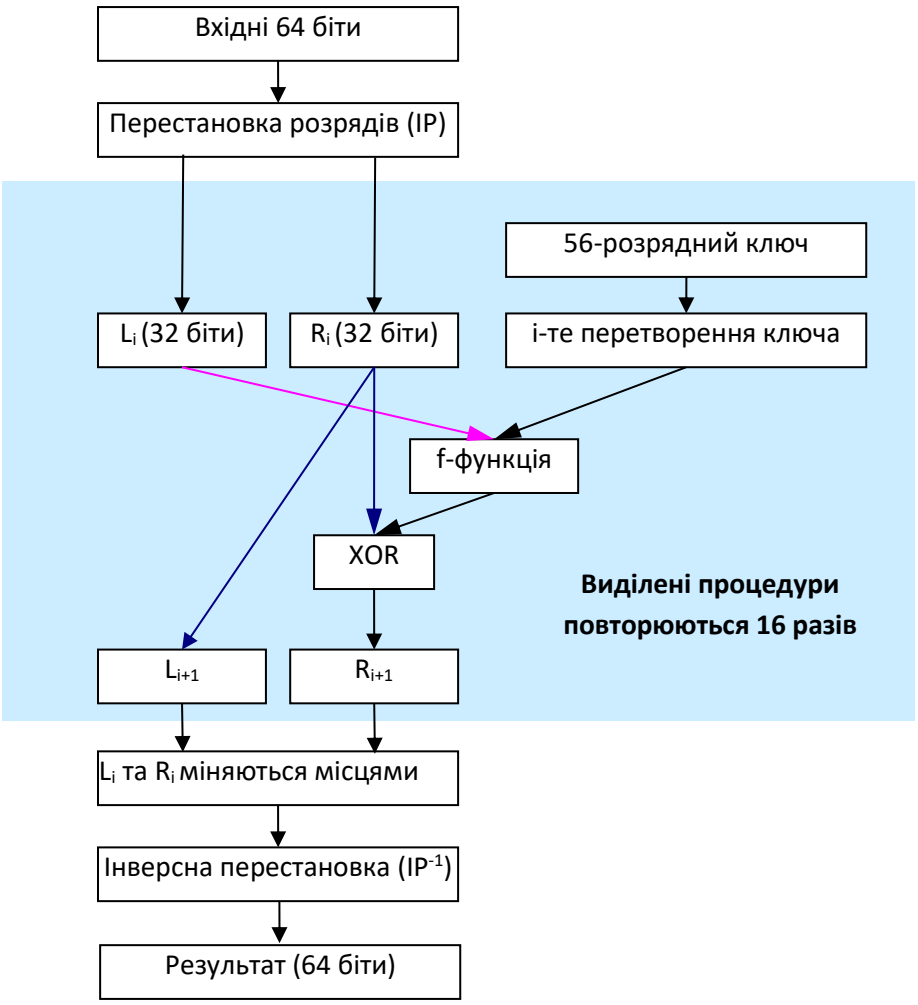

Рис. 2. Структурна схема реалізації алгоритму DES

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 і (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,…,16; K_n=KS(n, 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.

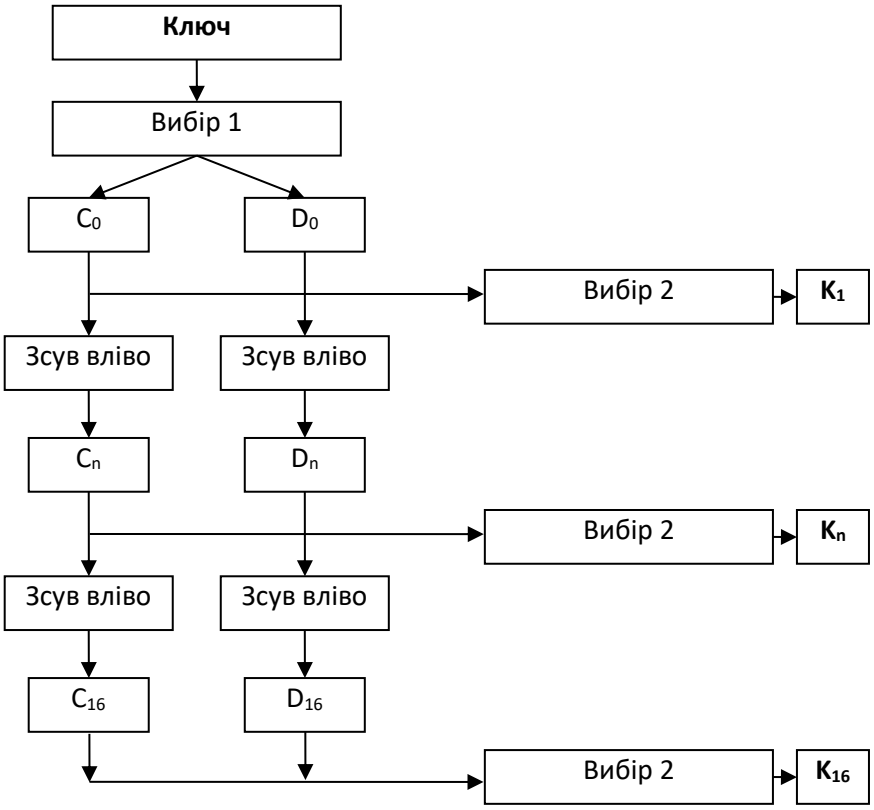

Рис. 3. Алгоритм обчислення послідовності ключів K_n

Таблиця 1. PC-1 (Вибір 1)

Заповнення C							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. PC-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 $\mathbf{b_m}$ у блоці даних. Перетворення полягає в заміні всіх елементів **m** на біти $\mathbf{b_s}$.

У C_0 увійдуть біти 57, 49, 41,..., 44 і 36, а в D0 - 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 P-блока.

P-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

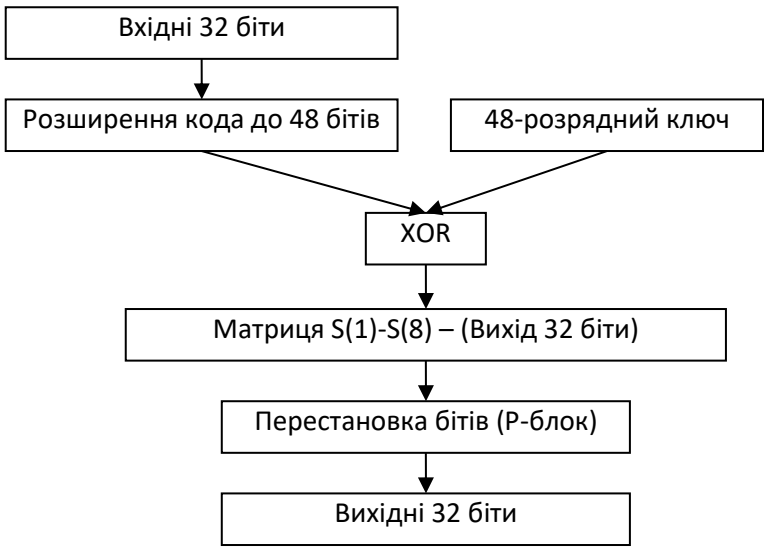

Рис. 4. Алгоритм роботи функції f

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 кожні 6 біт коду перетворяться 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 $К_1$, $К_2$, $К_3$, $К_4$, $К_5$,…, $К_{16}$, то ключами дешифровки будуть $К_{16}$, $К_{15}$, $К_{14}$, $К_{13}$, $К_{12}$,…, $К_1$). Це досягається тим, що зсув здійснюється не вліво, а вправо **(дивись табл. 3)**.

Проблеми DES

Так як довжина ключа дорівнює 56 бітів, існує 2^{56} 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 поводження 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.

Контрольна робота № 3

Тема: Асиметричні алгоритми шифрування. RSA.

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 $К_В$ по незахищеному каналу).

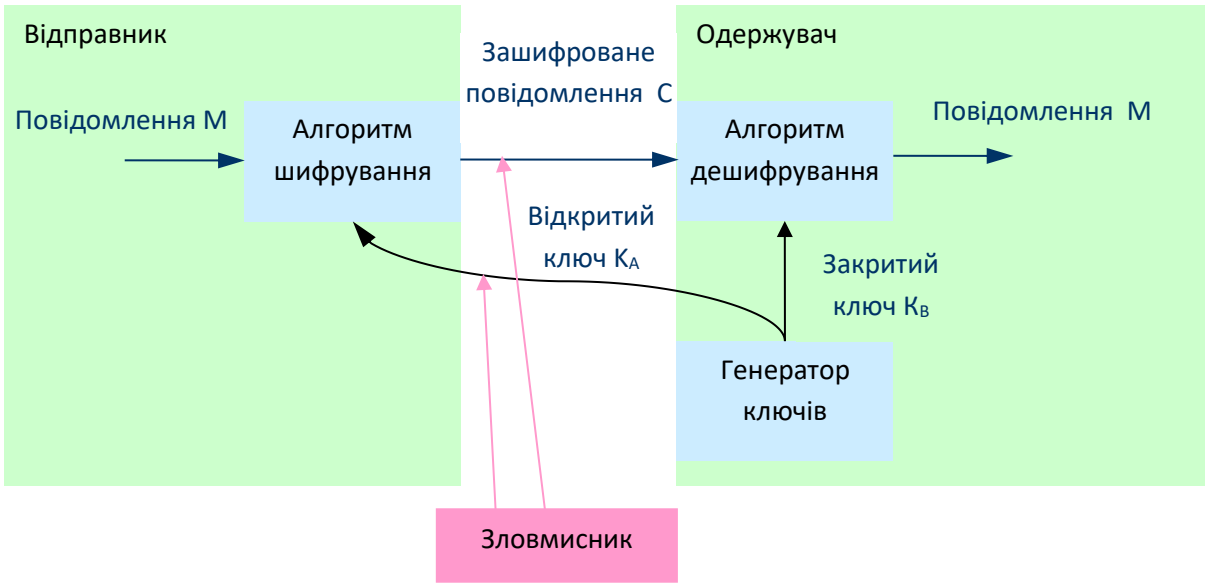

Рис.1. Узагальнена схема асиметричної криптосистеми з відкритим ключем.

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 (mod 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 <a, b> парою <b, r>, 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 (mod 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 (лаболаторна робота №2)

Контрольна робота № 4

Тема: Генератори псевдовипадкових чисел. Геш-функції.

Генератори псевдовипадкових чисел

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:

$$I(j+1)=a*I(j)(mod\ m)$$

При відповідному виборі констант. Константи запропоновані Park та Miller: $a=7^5=16807$, $m= 2^{31}-1=2147483647$ 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 Scharge 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 mod q) і r*[z/q] завжди лежать в інтервалі 0,..., m-1. Для множення (a*z)(mod 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 степінь 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 =am mod 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 ≠ 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 ≠ 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.

Геш-функції. MD5, SHA-2.

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 гешуванням 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 – геш 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 геш.

– 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 наявнсті двох вхідних

довжиною n. Ціми вхідними значеннями являються блок текста M, та геш-значення H_{i-1} попереднього блока текста (рис.1).

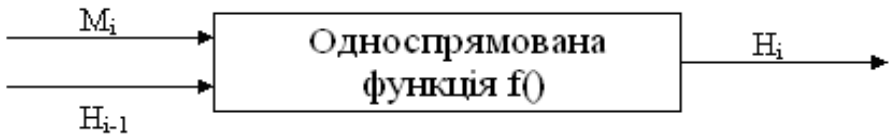

Рисунок 1 – Побудова односпрямованої геш-функції

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

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

https://uk.wikipedia.org/wiki/MD5

MD5 *(Message Digest 5)* – 128-бітний алгоритм гешування, 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]=HIGHEST_32_BITS(ABS(SIN(j+1))) j=0...63,

де функція HIGHEST_32_BITS(X) визначає 32 самих старших біта з двійкового запису дробового числа X, а операнд 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]) ROL s[j]

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

кінец_циклу

цикл по j **від** 16 **до** 31

T = (A + g(B,C,D) + x[z[j]] + y[j]) ROL s[j]

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

кінец_циклу

цикл по j **від** 32 **до** 47

T = (A + h(B,C,D) + x[z[j]] + y[j]) ROL s[j]

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

кінец_циклу

цикл по j **від** 48 **до** 63

T = (A + k(B,C,D) + x[z[j]] + y[j]) ROL s[j]

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

кінец_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 **and** V) **or** ((**not** U) **and** W).

– У другому раунді g(U,V,W)=(U **and** W) **or** (V **and** (not W)).

– У третьому раунді h(U,V,W)=U **xor** V **xor** W.

– У четвертому раунді k(U,V,W)=V **xor** (U **or** (**not** W)).

5) Підсумковий геш = ABCD (128 бітів)

SHA-2

https://uk.wikipedia.org/wiki/SHA-2

SHA-2 (англ. Secure Hash Algorithm Version 2 – безпечний алгоритм хешування, версія 2) – збірна назва односторонніх геш-функцій SHA-224,

SHA-256, SHA-384 і SHA-512. Геш-функції призначені для створення «відбитків» або «дайджестів» повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов'язаних із захистом інформації.

Загальний опис

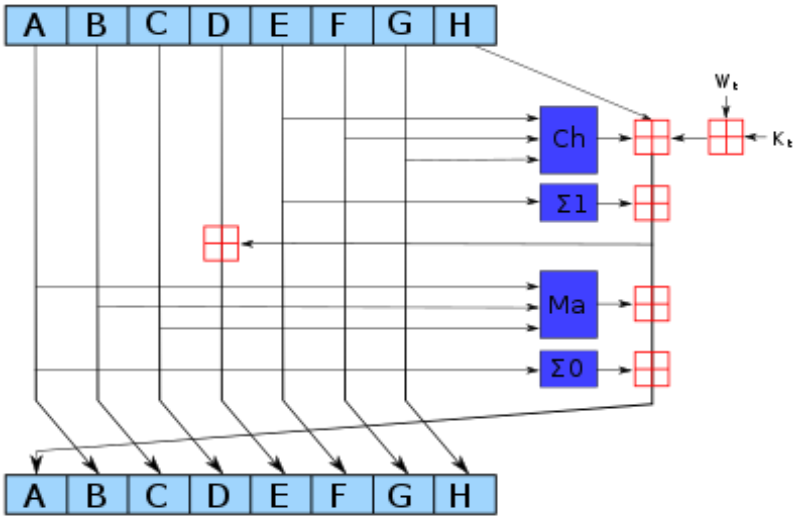

Рисунок 2 – Схема однієї ітерації алгоритмів SHA-2

Геш-функції сімейства *SHA-2* побудовані на основі структури Меркла-Демґарда.

Початкове повідомлення після доповнення розбивається на блоки, кожен блок – на 16 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64-ма чи 80-ма ітераціями (раундами). На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням геш-функції.

Алгоритм використовує такі бітові операції:

- ‖ – Конкатенація,
- + – Додавання,
- *And* – Побітове «І»,
- *Or* – Побітове «АБО»,

- *Xor* – Виключне «АБО»,
- *Shr* (Shift Right) – Логічний зсув вправо,
- *Rotr* (Rotate Right) – Циклічний зсув вправо.

У наступній таблиці показані деякі технічні характеристики різних варіантів SHA-2. «Внутрішній стан» означає проміжну геш-суму після обробки чергового блоку даних:

Геш-функція	**Довжина дайджесту повідомлення (біт)**	**Довжина внутрішнього стану (біт)**	**Довжина блоку (біт)**	**Максимальна довжина повідомлення (біт)**	**Довжина слова (біт)**	**Кількість ітерацій в циклі**
SHA-256/224	256/224	256	512	$2^{64} - 1$	32	64
SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80

SHA-3 (Keccak)

(https://en.wikipedia.org/wiki/SHA-3**)**

SHA-3 (*Keccak* – вимовляється як "кечак") – алгоритм ґешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Keccak став переможцем конкурсу криптографічних алгоритмів, що проводиться Національним інститутом стандартів та технологій США. 5 серпня 2015 року алгоритм затверджено та опубліковано як стандарт FIPS 202. У програмній реалізації автори заявляють про 12,5 цикли на байт при виконанні на ПК з процесором Intel Core 2. Однак в апаратних реалізаціях Keccak виявився набагато швидшим, ніж усі інші фіналісти.

Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak раніше).

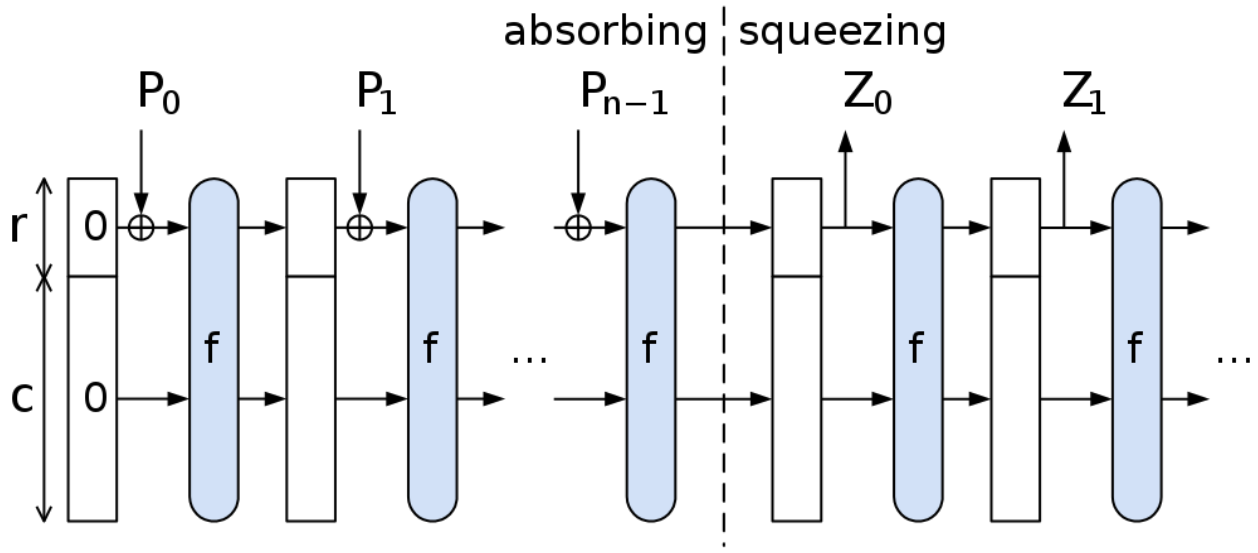

Рисунок 3 – Конструкція функції губки, використана в ґеш-функції. P_i – Вхідні блоки, Z_j – Вихід алгоритму. Розмір c («capacity») невикористовуваного для виведення набору бітів має бути значним для досягнення стійкості до атак

Алгоритм Keccak складається із трьох частин (рисунок 4).

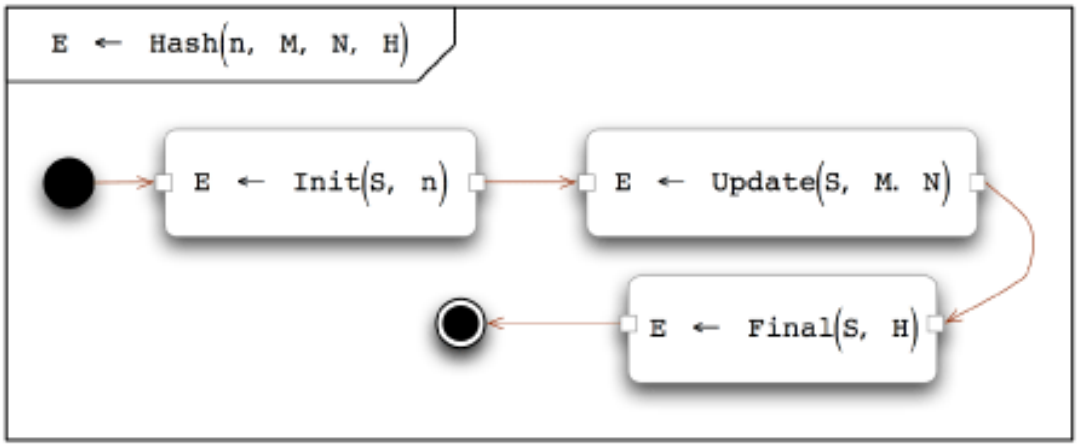

Рисунок 4 – Структура алгоритму SHA-3

Hash () використовується як функція входу. Для цього потрібні 4 вхідні параметри: розмір ґеш-біта (n), текст повідомлення (M) та його розмір у бітах (N), а також ґеш-змінна (H). Ця ґеш-змінна повинна мати наступну функцію malloc () Створіть:

```
HashReturn E;
char *H;
n = 224;
H = (char *)malloc(sizeof(char) * n / 8);
//...
E = Hash(n, M, N, H);
```

Функція Init () підготовляє внутрішній стан (S) для зазначеного розміру ґеша. Функція Update () починає стискати чи поглинати фазу. Тут текст повідомлення об'єднується відповідно до внутрішнього стану, а потім замінюється. Функція Final () починає вилучати або стискати фазу. Це ґеш-значення, витягнуте та зібране з внутрішнього стану. Зібраний ґеш із перших n бітів потім використовується як ґеш повідомлення. Усі чотири функції повертають результат помилки. Повернення 0 означає, що функція завершується без помилок.

На рисунку 5 показано структуру даних внутрішнього стану. Структура C з ім'ям spongeState містить 8 полів, два з яких є фіксованими масивами (відзначені червоним). У полі масиву станів зберігаються байти реального стану, а в полі масиву dataQueue зберігаються байти повідомлення, які будуть об'єднані та перетворені.

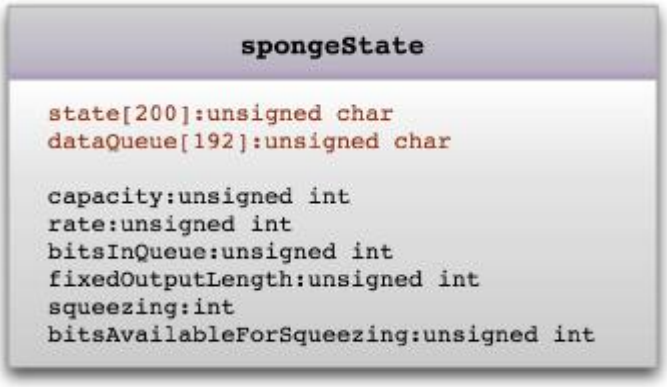

Рисунок 5 – Структура даних внутрішнього стану

Поле ємності – це ґеш-ємність (позначається c). Його значення встановлено вдвічі більше за розмір ґешу (2*n).

Поле швидкості – це довжина бітів повідомлення, оброблюваних у кожному циклі. Він встановлено значення 1600-c.

Поле bitInQueue – це довжина повідомлення в бітах, зарезервована у полі масиву dataQueue.

Поле fixedOutputlength – це бажаний розмір ґеша (позначається n).

Область стиснення – це символ режиму. Якщо встановлено значення 0, keccak перебуває у режимі стиснення. Якщо встановлено значення 1, keccak перебуває у режимі декомпресії. Крім того, файлові bitsAvailableForSqueezing зберігає довжину статусного біта, об'єднану в остаточний ґеш повідомлення.

Keccak використовує 24 цикли перетворення, щоб зменшити текст повідомлення до ґеш-значення. У кожному циклі безперервно викликаються п'ять модулів, як показано рисунку 6.

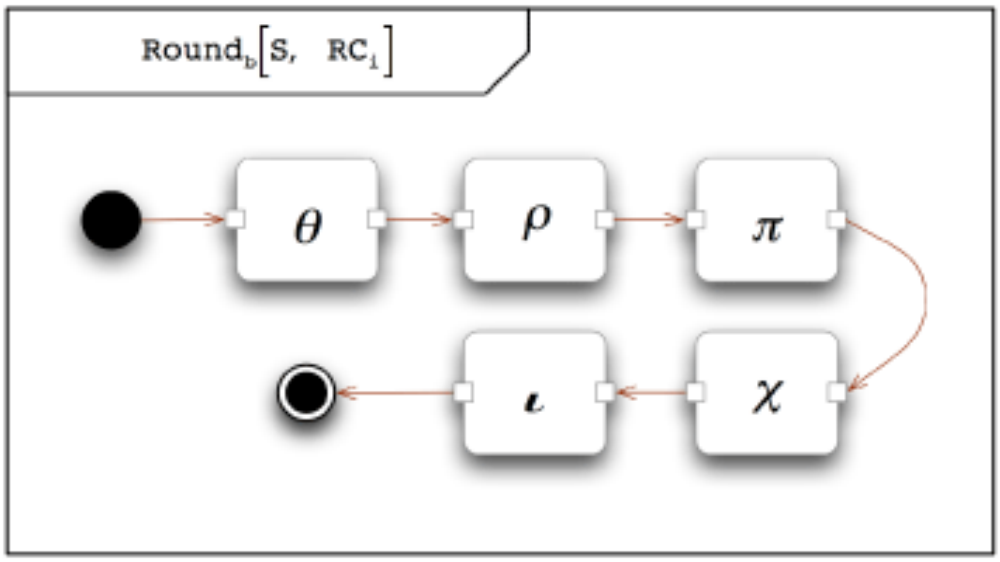

Рисунок 6 – Цикл перетворення SHA-3 (Keccak)

Модуль θ перетворює внутрішній стан масив 5x5, кожен елемент якого становить 64 біта. Він обчислює ту саму частину кожного стовпця, а потім використовує оператор виключає АБО (XOR) для їх об'єднання. Потім він виконує операцію XOR отриманого результату парності з кожним бітом стану наступним чином:

S[i][j][k] ^= parity(S[0...4][j-1][k])
^ parity(S[0...4][j+1][k-1])

where i = 0...4; j = 0...4; k = 0...63

Модуль ρ циклічно переміщає кожен 64-бітовий елемент відповідно до порядку трикутника. Але круговий рух виключає елемент S[0][0]. Модуль φ перетворює 64-бітові елементи. Перетворення слідує шаблону фіксованого призначення, показаному нижче:

S[j][2*i + 3*j] = S[i][j]

Модуль додає до циклу перетворення нелінійні характеристики. Він використовує лише три побітові оператори: AND, NOT і XOR для об'єднання елементів рядка. Потім запишіть результат у масив статусів таким чином:

S[i][j][k] ^= ~S[i][j + 1][k] & S[i][j + 2][k]

Модуль порушує будь-яку симетрію, створену іншими модулями. Він робить це шляхом XOR елемента у масиві з константою циклу. Цей модуль має 24 константи циклу для вибору. Ці константи визначені усередині Keccak.

Завдання: Розробити програму, яка реалізує алгоритми гешування MD5, SHA-2 та SHA-3. Порівняти швидкість виконання.

Контрольна робота № 5

Тема: Автентифікація користувача і електронний цифровий підпис.

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

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 и Q, потім знаходить їх добуток N=P*Q і значення функції φ(N)=(P-1)(Q-1).

Далі відправник обчислює число E з умов:

$$E \le \varphi(N),\ НСД\ (E, \varphi(N)) = 1$$

і число D з умов:

$$D < N,\ E * D \equiv 1\ (\mathrm{mod}\ \varphi(N)).$$

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 гешування 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 зловмисник 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 (лабораторна робота №3) та геш функції (лабораторна робота №4). **Розмір файлу не менш 100 кБ.**

Контрольна робота № 6

Тема: Криптоаналіз. Взлам RSA.

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, φ(n))=1 (тобто e та φ(n) – взаємопрості), де φ(n) – функція Ейлера від n.
3. З рівняння ed=1(mod φ(n)) знаходиться число d.

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

Процедура шифрування:

$$C=E_{(e,n)}(M)=M^e(\text{mod } n),$$

де C – зашифрований текст, M – відкритий текст, що задовольняє наступній умові: $M^{\varphi(n)}=1(\text{mod } n)$.

Процедура дешифрування:

$$M=D_{(d,n)}(C)=C^d(\text{mod } n).$$

Генерація цифрового підпису:

цифровий підпис $Q=M^d(\text{mod } n)$.

Перевірка цифрового підпису:

$$Q^e(\text{mod } n) \stackrel{?}{=} M.$$

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: $C^{e^J}(\mathrm{mod} n) = C$. Тобто супротивник просто здійснює j раз шифрування на відкритому ключі перехопленого шифротекста, це виглядає в такий спосіб:

$$(C^e)^e)^e..)^e(\mathrm{mod}\ n) = C^{e^J}(\mathrm{mod} n).$$

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

Приклад:

p=983, q=563, e=49, M=123456.

$C=M^{49}(\mathrm{mod}\ n)=1603$, $C^{497}(\mathrm{mod}\ n)=85978$,

$C^{498}(\mathrm{mod}\ n)=123456$, $C^{499}(\mathrm{mod}\ n)=1603$.

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, взаємопросте з N і обчислює $y=x^e(\mathrm{mod}\ n)$. 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

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

а значить $N^d=Sx^{-1}(\text{mod } n)$. 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

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

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

Супротивник обчислює

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

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 хеш-функцію.

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

Контрольна робота № 7

Тема: Програмні закладки. Клавіатурний шпигун. Стеганографія.

Програмні закладки

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.

За методом 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 виконуючими 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.

Клавіатурний шпигун

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 аутентифікацію користувачів. Перед виконанням замінювачам необхідно: 1) подібно комп'ютерному вірусу впровадитися в один або кілька системних файлів; 2) використовувати інтерфейсні зв'язку між програмними модулями системи аутентифікації 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 маскування слідів своєї присутності в системі.

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

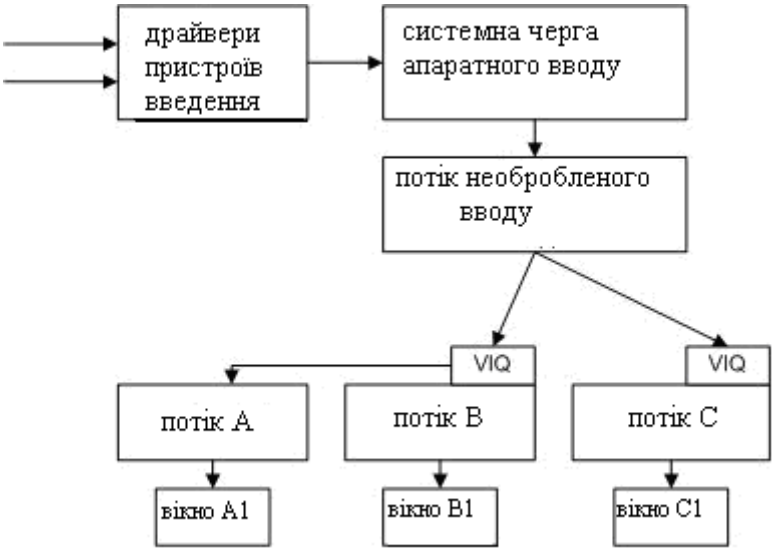

Рисунок 1 – Модель апаратного введення системи Windows

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'язане 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 напряму 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 USB);

– 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.

Завдання1:

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.

Стеганографія

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: к**ласична, 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.

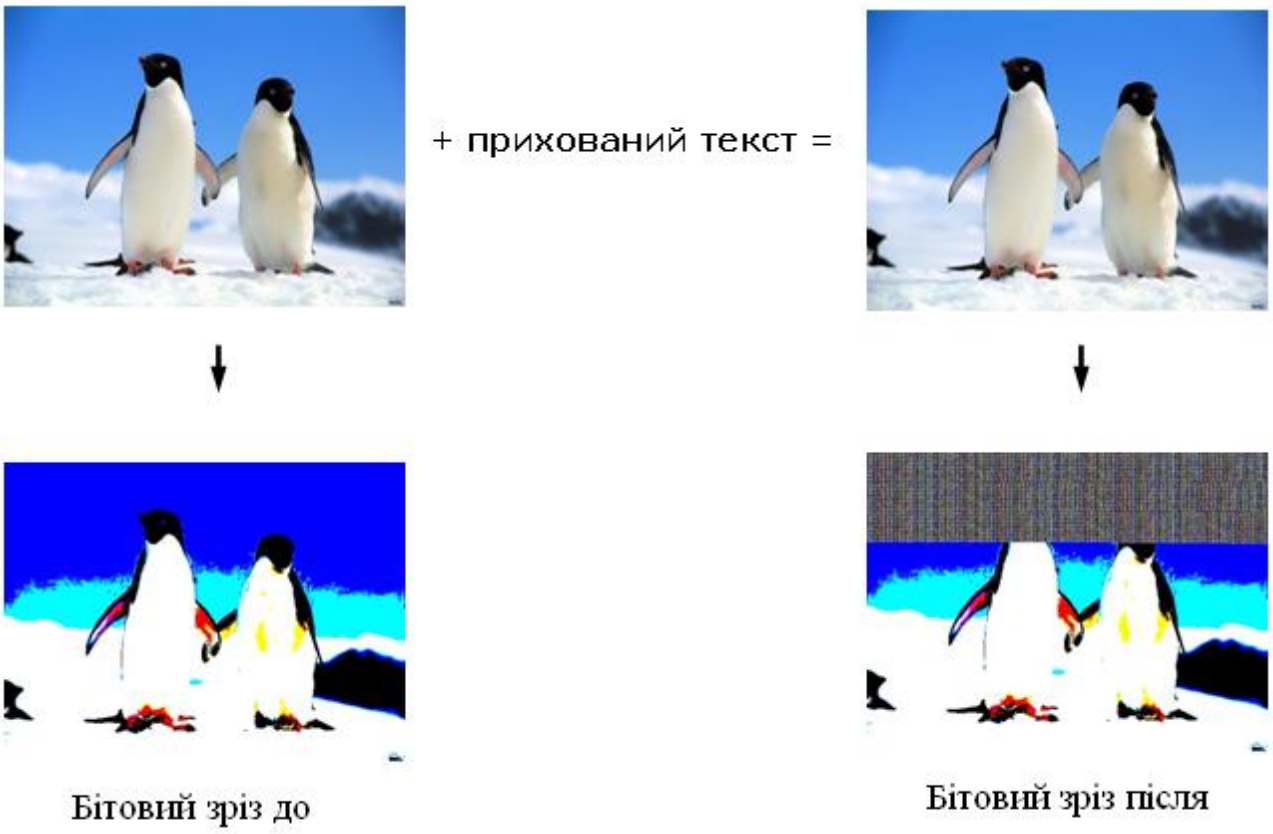

Рисунок 1 – Вигляд бітового зрізу файлу до та після вдубовування текстового повідомлення методом LSB

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 кольрів
Таблиця кольрів (ColorTable) (довжина змінюється від 8 до 1024 байт)			
ColorTable	4*N	x	палітра (256 елементів по 4 байтив для 256-кольрового зображення)
Дані зображення (BitMap Array)			
Image	Size	x	Зображення, записане по рядках з ліва на право і знизу вгору

ВАЖЛИВО!

1. Зображення зберігається порядково знизу-вгору. Для збереження кожного рядка виділяється кратна 4 кількість байт. У незначащих байтах зберігається сміття. Старшому біту або тетраді 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 кольрів; у цих файлах значення пікселів растрового масиву безпосередньо характеризують значення кольрів 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;
```

Завдання2:

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

Список використаної літератури

1. Смірнова Т.В., Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. –294 с. Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/9799

2. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

3. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

4. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації вавтоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

5. Смірнов О.А., Кавун С.В., Столбов В.Ф., Мелешко Є.В. Основи інформаційної безпеки. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. С.В. Кавуна. Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 26.04.2012 року № 1/11-5760. – Кіровоград: КНТУ 2012. – 442 с.

6. Смірнов О.А., Віхрова Л.Г., Осадчий С.І., Ковтун В.Ю., Мелешко Є.В. Основи захисту інформації. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.050201 «Системна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" 12 надано у відповідності з листом Міністерства освіти і науки України від 16.12.2010 року № 1/11-11486. – Кіровоград: КНТУ 2011. – 322 с.

7. Смірнов О.А., Кузнецов О.О., Євсеєв С.П., Мелешко Є.В., Король О.Г. Методи та алгоритми симетричної криптографії. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.О. Кузнецова. Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 26.04.2012 року № 1/11-5762. – Кіровоград: КНТУ 2012. – 315 с.

8. Захист інформації в автоматизованих системах управління : навчальний посібник / Уклад. І. А. Пількевич, Н. М. Лобанчикова, К. В. Молодецька. – Житомир : Вид-во ЖДУ ім. І. Франка, 2015. – 226 с.

9. Остапов С. Е. Технологія захисту інформації : навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 476 с.

10. Електронне урядування та електронна демократія: навч. посіб.: у 15 ч. / за заг. ред. А.І. Семенченка, В.М. Дрешпака. – К., 2017. Частина 13: Захист інформації в системах електронного урядування / [О.М. Хоша ба]. – К.: ФОП Москаленко О. М., 2017. – 72 с.

11. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 с.

12. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 с.

13. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 с.

14. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin.

CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 с.

15. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 с.

16. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 с.

Допоміжна

17. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399. (Scopus).

18. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications inComputer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184. (Scopus). Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85118101973&origin=AuthorNamesList&txGid=9fba77a9424db54ff3b099e4400c22bb13

19. Smirnova, T., Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and SystemsVolume 152, 2021, Pages 85-98. (Scopus). Режим доступу:https://www.scopus.com/record/display.uri?eid=2-s2.0-85090914783&origin=resultslist

20. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., «Метод розрахунку критичності галузевих інформаційно-телекомунікаційних систем». Наукоємні технології № 2(54), 2022. С. 94-104. Режим доступу: https://jrnl.nau.edu.ua/index.php/SBT/article/view/16757 (Фахове видання. Категорія «Б»)

21. Смірнова Т.В., Гнатюк С.О., Юдін О.Ю., Сидоренко В.М., Жаксигулова Д.Д., «Експериментальне дослідження моделі розрахунку кількісного критерію оцінювання захищеності інформаційно-телекомунікаційних систем критичної інфраструктури держави» Кібербезпека: освіта, наука, техніка. № 4(16). 2022. С. 6-18. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/359/298 (Фахове видання. Категорія «Б»)

22. Смірнова Т.В., Якименко Н.М., Смірнов О.А., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Вісник Хмельницького національного університету. Серія: «Технічні науки», № 2 (307). С. 46-52. 2022. Режим доступу: http://journals.khnu.km.ua/vestnik/?cat=65 (Фахове видання. Категорія «Б»)

23. Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., Смірнов О.А. «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 1(67). С. 84-89. Режим доступу: http://journals.nupp.edu.ua/sunz/article/view/2449/1918 (Фахове видання. Категорія «Б»)

24. Смірнова Т.В., Якименко Н.М., Улічев О.С., Конопліцька-Слободенюк О.К., Смірнов С.А., «Дослідження лінійних перетворень запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Кібербезпека: освіта, наука, техніка. № 3(15). С. 85-92. 2022. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/337 (Фахове видання. Категорія «Б»)

25. Смірнова Т.В., Бурмак Ю.А., Улічев О.С., Усік П.С., Доренський О.П., «Стійка функція шифрування удосконаленого модуля криптографічного захисту інформації в інформаційно-комунікаційних системах» Кібербезпека: освіта, наука, техніка. № 1(13). С. 183-201. 2021. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/346 (Фахове видання. Категорія «Б»)

26. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Бурмак Ю.А., Оспанова Д.М., «Удосконалений модуль криптографічного захисту інформації в сучасних інформаційно-комунікаційних системах та мережах».

Кібербезпека: освіта,наука, техніка. № 2(14). С. 176-185. 2021. Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/329 (Фахове видання. Категорія «Б»)

27. Смірнова Т.В., Смірнов О.А., Смірнов С.А., Поліщук Л.І., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87. https://doi.org/10.28925/2663-4023.2019.3.6387 Режим доступу: http://nbuv.gov.ua/UJRN/cest_2019_3_7 (Фахове видання).

28. Смірнова Т.В., Смірнов О.А., Смірнов С.А., Поліщук Л.І., КонопліцькаСлободенюк О.К.,. GERT-моделі технології хмарного антивірусного захисту. Кібербезпека: освіта, наука, техніка. – Том 2 № 2. – Київ: КУ ім.. Бориса Грінченка. – 2018. – С. 7-30. https://doi.org/10.28925/2663-4023.2018.2.730 Режим доступу: http://nbuv.gov.ua/UJRN/cest_2018_2_3 (Фахове видання).