ISSN 2519-2310

CS&CS Journal

KARAZIN UNIVERSITY
CLASSICS AHEAD OF TIME

2(18) 2020

COMPUTER SCIENCE AND CYBERSECURITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В.Н.КАРАЗІНА
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ имени В.Н. КАРАЗИНА
V.N. KARAZIN KHARKIV NATIONAL UNIVERSITY

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА
КОМПЬЮТЕРНЫЕ НАУКИ И КИБЕРБЕЗОПАСНОСТЬ
COMPUTER SCIENCE AND CYBERSECURITY
(CS&CS)

Issue 2(18) 2020

Заснований 2015 року

Міжнародний електронний науково-теоретичний журнал
Международный электронный научно-теоретический журнал
International electronic scientific journal

The journal publishes research articles on theoretical, scientific and technical problems of effective facilities development for computer information communication systems and on information security problems based on advanced mathematical methods, information technologies and technical means.

Journal is published quarterly.
Approved for placement on the Internet by Academic Council of the Karazin Kharkiv National University (December 28, 2020, protocol No.19)

The journal has Digital Object Identifier: **10.26565/2519-2310**.

Editorial office:
V.N. Karazin Kharkiv National University
Svobody Sq., 6, office 315a, Kharkiv, 61022, Ukraine *(North building of University, 3th floor)*
Phone: +38 (057) 705-10-83
E-mail: *cscsjournal@karazin.ua*
Web-page: *http://periodicals.karazin.ua/cscs (Open Journal System)*

Published articles have been internally and externally peer reviewed

TABLE OF CONTENTS

Issue 2(18) 2020

Знаходження оптимального VPN рішення
на основі методу аналізу ієрархій ……..... 4
Н. Усіченко, С. Рузудженк, К. Погоріла

Огляд статичних методів аналізу
зловмисного програмного забезпечення ……..……..... 15
О. Прищепа, О. Доценко

Верифікація відбитків пальців з
використанням рішення задачі комівояжера
і декомпозиції оточення мінуцій 25
О. Мелкозерова, С. Малахов, В. Гайкова

Спосіб криптологічних перетворень даних ..……..……..... 33
М. Сукнов, І. Громико, Є. Перчик

УДК 004.7: 004.7(075) DOI: 10.26565/2519-2310-2020-2-01

ЗНАХОДЖЕННЯ ОПТИМАЛЬНОГО VPN РІШЕННЯ НА ОСНОВІ МЕТОДУ АНАЛІЗУ ІЄРАРХІЙ

Нікіта Усіченко, Сабіна Рузудженк, Каріна Погоріла

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
nik.usichenko@gmail.com, ruzudzhenk.jb@gmail.com, karina.pogorelka@gmail.com

Рецензент: Роман Олійников, д.т.н., проф., ПАО "ІІТ", вул. Бакулина, 12, Харків, 61166, Україна
roliynykov@gmail.com

Поступила: Червень 2020

Аннотация. *В роботі розглянуто проблематику щодо створення віртуальної приватної мережі в сучасному світі. Запропоновано метод знаходження оптимального рішення віртуальної приватної мережі на основі методу аналізу ієрархій. Наведені переваги та недоліки даного методу. Через доступність широкого спектру програмно-апаратних засобів, розглянуті рішення на основі п'яти актуальних VPN протоколів: PPTP, IPsec, L2TP + IPsec, SSTP та OpenVPN. Залученими до виконання досліджень експертами введені шість критеріїв для визначення кращого VPN рішення. Введені наступні критерії: швидкість, шифрування даних, налаштування, конфігурація, порти, стабільність та сумісність клієнтів. Надана ієрархія рівнів прийняття рішень для тематичного дослідження. Виконана оцінка пріоритетів на підставі експертних суджень, профільних, за даної тематики, спеціалістів. Виконана перевірка узгодженості для виявлення можливих суперечностей. Визначений глобальний пріоритет за допомогою методу власних значень, який обчислює не тільки пріоритети, але й ступінь невідповідності. За результатами моделювання підкреслено, що вибір VPN протоколу є складним завданням. Вирішення цього завдання потребує аналізу ринку, визначення критеріїв порівняння і надання пріоритетів. Звернено увагу на той факт, що всі зазначені складові здійснюються в умовах, коли немає повної інформації о системі, в якій ці процеси відбуваються. В такому випадку, необхідно користуватися методами прийняття рішень в умовах невизначеності. На теперішній час існує велика кількість таких методів, але в ситуації, що розглядалася, експертами було запропоновано метод аналізу ієрархій. За умов наведених критеріїв та наданих пріоритетів, за результатами розрахунку, протокол OpenVPN являється найбільш оптимальним рішенням для створення віртуальної приватної мережі.*

Ключевые слова*: метод аналізу ієрархій; віртуальна приватна мережа; метод Саати; VPN протокол.*

1 Вступ

На думку аналітиків обсяг світового ринку віртуальних приватних мереж оцінювався у 25,65 млрд. доларів США у 2019 році та, як очікується, буде рости із середньорічним темпом зростання (CAGR) 17,4% з 2020 по 2027 рік [1]. Такий інтерес до VPN (Virtual private network) обумовлений необхідністю зниження витрат на обслуговування корпоративних мереж за допомогою з'єднання віддалених користувачів через Інтернет. При цьому, головним питанням при організації такого типу з'єднань стає безпека передачі даних, що спонукає до створення механізмів, за допомогою яких можна забезпечити цілісність і конфіденційність інформації, яка циркулює між віддаленими користувачами.

Основним інструментом для розв'язання такого питання може виступати VPN. Найголовнішою перевагою цього рішення є фінансова сторона питання. В теперішній час, коли панує пандемія і впроваджуються масові карантини заходи, коли усе більше компаній змушені переводити своїх співробітників на дистанційну роботу, стає актуальним питання ефективності систем безпечного доступу на основі VPN [2].

За останні роки було реалізовано велику кількість різних варіантів організації віртуальної приватної мережі [3]. Через це перед користувачами, для яких використання віддаленого доступу до корпоративних ресурсів є основним варіантом підтримки бізнес процесів, стає *завдання* вибору найбільш відповідного (*для конкретних умов підтримки бізнес процесів*) рішення для VPN на основі певних пріоритетів. Ця *задача* може вирішуватися за допомогою теорії прийняття рішень.

В області теорії прийняття рішень за останні роки опубліковано багато наукових робіт, присвячених як вибору варіантів при створенні складних технічних систем, так і, безпосередньо, розвитку методів теорії прийняття рішень. Серед них можна відзначити роботи, в яких

викладено методи теорії прийняття рішень, що включають методи аналізу ієрархій та аналітичних мереж, методи, засновані на теорії нечітких множин тощо [4-6].

Так наприклад, при проектуванні нових зразків техніки або оцінці її досяжного технічного рівня, завжди виникає питання доцільності використання того чи іншого методу прийняття рішень. З огляду на актуальність зазначеного завдання було проведено цикл досліджень стосовно оцінки актуальних загроз безпечного доступу для IT компаній та синтезу відповідних рішень задля підтримки безпечного обміну корпоративної інформацією, як в сегменті головний офіс – працівник, так і на ділянках працівник – працівник. Варто зазначити, що в межах виконання досліджень в якості експертів залучалися профільні фахівці відповідної кафедри ХНУ імені В.Н. Каразіна.

2 Основні концепції методу аналізу ієрархій

Для умов невизначеності, коли немає можливості однозначно порівняти альтернативи між собою, існують наступні методи прийняття рішень: *Multi-Attribute Utility Theory* [7], *SMART* [8], *ELECTRE* [9], та Метод аналізу ієрархій (MAI) [10].

MAI був розроблений Томасом Сааті, та дозволяє структурувати проблему прийняття рішень у вигляді ієрархії, порівняти і зробити оцінку альтернативних варіантів рішень. В порівнянні з іншими методами MAI має наступні переваги:

- може враховувати відносні пріоритети факторів або альтернатив і представити найкращий результат;
- забезпечує просту і дуже гнучку модель для проблеми;
- забезпечує можливість перерахувати або структурувати на будь-якому рівні деталізації основної проблеми;
- забезпечує можливість вимірювати узгодженість суджень [11].

Водночас цьому методу притаманні певні недоліки:

- MAI має суб'єктивний характер процесу моделювання, тобто методологія не може гарантувати прийняття рішень, як однозначно істинних;
- коли кількість рівнів в ієрархії збільшується, кількість парних порівнянь також збільшується, так що, для побудови моделі MAI потрібно набагато більше часу та зусиль [11].

Для використання MAI користувачеві потрібно виконати два кроки, щоб отримати рейтинг альтернатив: спершу, як і у випадку з будь-яким іншим методом прийняття рішень, проблема повинна бути структурована, після цього бали - або пріоритети, як вони відомі в MAI - обчислюються на основі попарних порівнянь, наданих користувачем. Можна здійснити два додаткові етапи: перевірку узгодженості та аналіз чутливості. Обидва кроки є необов’язковими, але рекомендуються як підтвердження надійності результатів. Перевірка узгодженості є загальною для всіх методів, заснованих на попарних порівняннях. Проблеми, що вимагають техніки багатокритеріального прийняття рішень, є складними, тому їх зручно розподіляти на кілька підзадач. Розподіл здійснюється під час структурування проблеми та визначення пріоритетів шляхом попарного порівняння. Проблема структурується відповідно до ієрархії, де верхній елемент є метою рішення, другий рівень ієрархії представляє критерії, а найнижчий - альтернативи. У складніших ієрархіях можна додати більше рівнів. Ці додаткові рівні представляють підкритерії. У будь-якому випадку в ієрархії є мінімум три рівні.

3 Застосування методу аналізу ієрархій для вибору оптимального VPN рішення

Експерти, що приймали участь у моделюванні, обирали віртуальну приватну мережу на основі п’яти наступних VPN протоколів через те, що для них доступний широкий спектр програмно-апаратних засобів:

- *PPTP (Point-to-Point tunneling protocol);*
- *IPsec (IP Security);*
- *L2TP (Layer Two Tunneling Protocol) + IPsec (Internet Protocol Security);*
- *SSTP (Secure Socket Tunneling Protocol);*

- *OpenVPN.*

При прийнятті рішення щодо вибору VPN були визначені наступні шість критеріїв:

- швидкість (*обсяг даних, оброблених і переданих за одиницю часу*);
- шифрування даних при передаванні відкритими мережами;
- налаштування / конфігурація (*складність налаштування*);
- порти (*можливість блокування*);
- стабільність (*стабільність з'єднання та швидкість відновлення роботи*);
- сумісність клієнтів (*кросплатформність*).

На рис. 1 представлено ієрархію тематичного дослідження. Він містить три рівні, мінімально необхідних для вирішення проблеми з MAI, де кожен нижній рівень має пріоритет відповідно до його безпосереднього верхнього рівня. Відповідне питання щодо розставлення пріоритетів залежить від контексту, а іноді, й від того, хто приймає рішення.

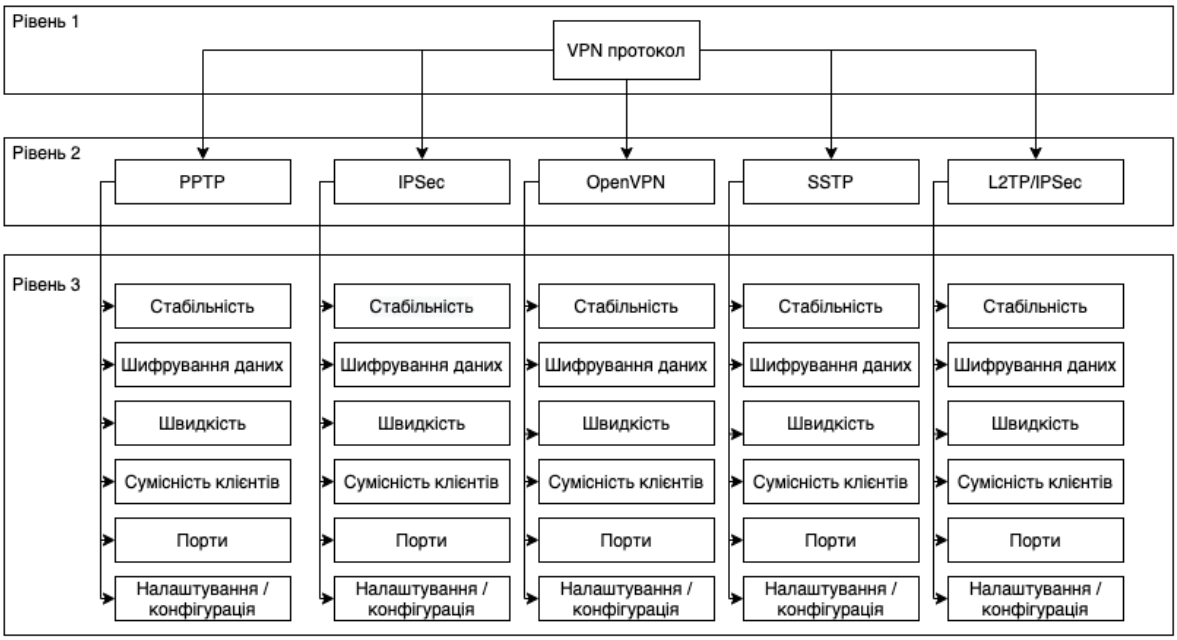


Рис. 1 - Ієрархія рівнів прийняття рішень для тематичного дослідження

Наступним кроком є визначення пріоритету, де оцінки пріоритетів формувались на основі експертних суджень спеціалістів, що залучались до експерименту (*фахівці кафедри*). При цьому пріоритетом є оцінка, яка визначає важливість альтернативи або критерію в рішенні. Після етапу структурування проблем необхідно розрахувати три наступні типи пріоритетів.

- *Критерії пріоритетів*. Важливість кожного критерію (стосовно вищої мети).
- *Локальні альтернативні пріоритети*. Важливість альтернативи щодо одного конкретного критерію.
- *Глобальні альтернативні пріоритети*. Критерії пріоритетності та локальні альтернативні пріоритети - це проміжні результати, що використовуються для розрахунку глобальних альтернативних пріоритетів. Глобальні альтернативні пріоритети класифікують альтернативи за всіма критеріями, а отже, і загальною метою.

Використання парних порівнянь, як правило, оцінюється за фундаментальною шкалою 1–9. Перетворення з вербальної в числову шкалу наведено в таблиці 1.

Таблиця 1 – Таблиця перетворення

Ступінь важливості	Визначення
1	Рівне значення
2	Слабка перевага
3-4	Помірна перевага
5	Сильна перевага
6-7	Дуже сильна перевага
8-9	Надзвичайна перевага

У таблиці 2 наведені попарні порівняння між критеріями. Всі порівняння є додатними числами. Порівняння по головній діагоналі дорівнюють 1, оскільки критерій порівнюється сам із собою. Матриця є взаємною, оскільки верхній трикутник є реверсом нижнього трикутника. Перевага точності вимагає більших зусиль, особливо коли існує велика кількість критеріїв або альтернатив. Кількість необхідних порівнянь для кожної матриці порівняння дорівнює $(n^2-n)/2$, де n - кількість альтернатив / критеріїв.

Таблиця – 2 (Критерії)

Критерії	*Стабільність*	*Шифрування даних*	*Швидкість*	*Сумісність клієнтів*	*Порти*	*Налаштування*
Стабільність	1	1	2	3	5	6
Шифрування даних	1	1	2	3	5	6
Швидкість	1/2	1/2	1	2	3	5
Сумісність клієнтів	1/3	1/3	1/2	1	2	4
Порти	1/5	1/5	1/3	1/2	1	3
Налаштування	1/6	1/6	1/5	1/4	1/3	1

Це значення отримано наступним образом:

- n^2 - загальна кількість порівнянь, які можна записати в матрицю;
- n з них представляють порівняння альтернативи із самою собою (головна діагональ). Ця оцінка дорівнює 1 і тому не потрібна;
- оскільки матриця є взаємною, потрібна лише половина порівнянь. Інша половина автоматично обчислюється з першої половини.

У таблицях 3 - 8 наведені матриці розрахунку пріоритетів для наступних критеріїв: *налаштування, швидкість, порти, шифрування даних, стабільність, сумісність клієнтів*.

Таблиця – 3 (Налаштування / конфігурація)

	OpenVPN	*L2TP/IPsec*	*IPsec*	*SSTP*	*PPTP*
OpenVPN	1	1/3	1/4	2	1/6
L2TP/IPsec	3	1	1/2	4	1/3
IPsec	4	2	1	5	2
SSTP	1/2	1/4	1/5	1	1/7
PPTP	6	3	1/2	7	1

Таблиця – 4 (Швидкість)

	OpenVPN	*L2TP/IPsec*	*IPsec*	*SSTP*	*PPTP*
OpenVPN	1	3	2	1	1/2
L2TP/IPsec	1/3	1	1/2	1/3	1/6
IPsec	1/2	2	1	1/2	1/4
SSTP	1	3	2	1	1/2
PPTP	2	6	4	2	1

Таблиця – 5 (Порти)

	OpenVPN	*L2TP/IPsec*	*IPsec*	*SSTP*	*PPTP*
OpenVPN	1	4	3	1	7
L2TP/IPsec	1/4	1	1/2	1/4	1/5
IPsec	1/3	2	1	1/3	3
SSTP	1	4	3	1	7
PPTP	1/7	5	1/3	1/7	1

Таблиця – 6 (Шифрування даних)

	OpenVPN	*L2TP/IPsec*	*IPsec*	*SSTP*	*PPTP*
OpenVPN	1	1	3	1	6
L2TP/IPsec	1	1	3	1	6
IPsec	1/3	1/3	1	1	3
SSTP	1	1	1	1	6
PPTP	1/6	1/6	1/3	1/6	1

Таблиця – 7 (Стабільність)

	OpenVPN	*L2TP/IPsec*	*IPsec*	*SSTP*	*PPTP*
OpenVPN	1	1	1	1	2
L2TP/IPsec	1	1	1	1	2
IPsec	1	1	1	1	2
SSTP	1	1	1	1	2
PPTP	1/2	1/2	1/2	1/2	1

Таблиця – 8 (Сумісність клієнтів)

	OpenVPN	L2TP/IPsec	IPsec	SSTP	PPTP
OpenVPN	1	1/3	1/3	4	1/3
L2TP/IPsec	3	1	1	7	1
IPsec	3	1	1	7	1
SSTP	1/4	1/7	1/7	1	1/7
PPTP	3	1	1	7	1

Коли матриця завершена, виконується перевірка узгодженості для виявлення можливих суперечностей у записах. Коли подано кілька послідовних попарних порівнянь, вони можуть суперечити одне одному. Причинами цих суперечностей можуть бути, наприклад, нечітко визначені проблеми, відсутність достатньої кількості інформації, невизначена інформація або відсутність концентрації.

У більшості випадків проблеми визначені нечітко [10]. Той, хто приймає рішення, може мати досить поверхневе уявлення про бажання створити віртуальну приватну мережу, при цьому не знаючи точних альтернатив та критеріїв. Структура повинна бути сформована шляхом тривалих розмірковувань, аналізу досвіду і організації спеціалізованих груп тощо.

Таке структурування елементів рішення є важливим, оскільки інша структура може призвести до різного остаточного рейтингу. Критерії з великою кількістю підкритеріїв, як правило, набувають більшої ваги, ніж тоді, коли вони менш деталізовані.

MAI, завдяки його попарним порівнянням, потребує шкали співвідношень, які, на відміну від методів, що використовують інтервальні порівняння, не потребують одиниць порівняння. Судження є відносною величиною або часткою *a / b* двох величин «a» та «b», що мають однакові одиниці виміру: - інтенсивність, корисність тощо.

Матриця, заповнена попарним порівнянням a_{ij}, називається узгодженою, якщо дотримуються правила транзитивності та взаємності [10].

Правило транзитивності: $a_{ij} = a_{ik} \times a_{kj}$, де a_{ij} - порівняння альтернативи i з j.

Правило взаємності: $a_{ji} = 1 / a_{ij}$, де i, j та k - будь-які альтернативи матриці.

Якщо ми припустимо, що переваги P_i відомі,

$$A = \begin{bmatrix} p_1/p_1 & \cdots & p_i/p_1 & \cdots & p_i/p_1 \\ \cdots & 1 & \cdots & \cdots & \cdots \\ p_i/p_1 & \cdots & 1 & \cdots & p_i/p_1 \\ \cdots & \cdots & \cdots & 1 & \cdots \\ p_n/p_1 & \cdots & p_n/p_j & \cdots & p_i/p_1 \end{bmatrix},$$

то можна створити цілком узгоджену матрицю, оскільки всі порівняння a_{ij} підпорядковуються рівності: $a_{ji} = P_i / P_j$, де P_i є пріоритетом альтернативи i.

Пріоритети відношень в MAI заздалегідь не відомі. Оскільки пріоритети відношень мають сенс лише у випадку, якщо вони походять з узгоджених або майже узгоджених матриць, необхідно застосовувати перевірку узгодженості. Поріг для визначення непослідовної матриці незрозумілий. Для вимірювання узгодженості існує кілька методів. Один з них заснований на визначнику матриці, інший - на додаванні різниці між співвідношенням розрахованих пріоритетів та наведеними порівняннями. Однак, найчастіше, використовується метод, розроблений Т. Сааті [10], який запропонував *індекс узгодженості* (*CI*), що пов'язаний із методом власних значень:

$$CI = \frac{\lambda_{max} - n}{n - 1},$$

де λ_{max} - максимальне власне значення. *Відношення узгодженості* (*CR*) визначається як: $CR = CI / RI$, де *RI* - випадковий індекс (*середній CI 500 випадково заповнених матриць*). Якщо CR менше ніж 10% (*невідповідність менше ніж 10% з 500 випадково заповнених матриць*), то матриця має прийнятну узгодженість.

Сааті розрахував випадкові показники (*див. в Табл. 9*), а інші дослідники проводили моделювання з різною кількістю матриць або неповними матрицями. Їх випадкові індекси декілька відрізняються, але близькі до показників Сааті.

Таблиця – 9 (Випадкові показники Сааті)

n	3	4	5	6	7	8	9	10
RI	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

Розрахуємо *CI* та *CR* для всіх матриць:

Таблиця – 10 (Критерії)

Критерії	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
Стабільність	180,00	2,100	0,2865	0,92
Шифрування даних	180,00	2,100	0,2865	0,92
Швидкість	7,5000	1,334	0,1819	1,10
Сумісність клієнтів	0,4444	0,891	0,1215	1,18
Порти	0,0200	0,572	0,0780	1,27
Налаштування	0,0005	0,334	0,0456	1,14

Індекс узгодженості для таблиці 10 дорівнює: $CI = 0,11$. З цього слід, що відношення узгодженості дорівнює: $CR = CI / RI = 8,54\ \%$ – А так як $CR < 10\%$ - матриця 10 має прийнятну узгодженість.

Таблиця – 11 (Налаштування / конфігурація)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	0,03	0,49	0,07	1,06
L2TP/IPSec	2,00	1,15	0,17	1,14
IPSec	80,00	2,40	0,36	0,88
SSTP	0,004	0,32	0,05	0,93
PPTP	63,00	2,29	0,34	1,25

Індекс узгодженості для табл. 11: $CI = 0,07$.

Відношення узгодженості дорівнює: *CR = CI / RI = 7,62 %*
CR < 10% - матриця 11 має прийнятну узгодженість.

Таблиця – 12 (Швидкість)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	3,00	1,25	0,21	1,00
L2TP/ IPsec	0,01	0,39	0,06	0,97
IPsec	0,13	0,66	0,11	1,04
SSTP	3,00	1,25	0,21	1,00
PPTP	96,00	2,49	0,41	1,00

Індекс узгодженості для табл. 12: *CI = 0,0016*.
Відношення узгодженості: *CR = CI / RI = 0,14 %*
CR < 10% - матриця 12 має прийнятну узгодженість.

Таблиця – 13 (Порти)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	84,00	2,43	0,37	1,00
L2TP/ IPsec	0,01	0,36	0,05	0,87
IPsec	0,67	0,92	0,14	1,09
SSTP	84,00	2,43	0,37	1,00
PPTP	0,03	0,51	0,08	1,39

Індекс узгодженості для табл. 13: *CI = 0,09*.
Відношення узгодженості: *CR = CI / RI = 7,66 %*
CR < 10 % - матриця 13 має прийнятну узгодженість.

Таблиця – 14 (Шифрування даних)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	18,00	1,78	0,29	1,03
L2TP/ IPsec	18,00	1,78	0,29	1,03
IPsec	0,33	0,80	0,13	1,10
SSTP	6,00	1,43	0,24	0,98
PPTP	0,002	0,27	0,05	0,99

Індекс узгодженості: *CI = 0,03*.
Відношення узгодженості: *CR = CI / RI = 3 %*
CR < 10 % - матриця 14 має прийнятну узгодженість.

Таблиця – 15 (Стабільність)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	2,00	1,15	0,22	1
L2TP/IPsec	2,00	1,15	0,22	1
IPSec	2,00	1,15	0,22	1
SSTP	2,00	1,15	0,22	1
PPTP	0,06	0,57	0,11	1

Індекс узгодженості: *CI = 0,00*.

Відношення узгодженості: *CR = CI / RI = 0%*
CR < 10% - матриця 15 має прийнятну узгодженість.

Таблиця – 16 (Сумісність клієнтів)

	Добуток	Оцінки компонент власного вектору	Нормалізовані оцінки вектору пріоритету	L_{max}
OpenVPN	0,15	0,68	0,11	1,09
L2TP/ IPsec	21,00	1,84	0,29	0,99
IPsec	21,00	1,84	0,29	0,99
SSTP	0,001	0,24	0,04	0,95
PPTP	21,00	1,84	0,29	0,99

Індекс узгодженості: *CI = 0,01*.
Відношення узгодженості: *CR = CI / RI = 0,5%*
CR < 10% - матриця 16 має прийнятну узгодженість.

Визначення глобальних пріоритетів є завершальною стадією MAI. Існують різні методи для розрахунку пріоритетів за допомогою матриці попарного порівняння, найпопулярніший - це *метод власних значень* [10]. Метод власних значень обчислює не тільки пріоритети, але й ступінь невідповідності. У методі власних значень вектор пріоритетів *p* обчислюється вирішенням рівняння: *Ap = np* , де *n* - розмірність A і $p = (p_1, \ldots, p_j, \ldots, p_n)$.

Спочатку продемонстровано обґрунтованість методу власних значень на послідовній матриці. Припустимо, що пріоритети відомі. Легко вивести послідовну матрицю порівняння з пріоритетів наступним чином. Нехай $a_{ij} = p_i / p_j$. Помноживши *A* на вектор пріоритету *p*, отримаємо праву частину рівняння. Для спрощення обчислення спочатку розглядається лише рядок *i* з *A*:

$$\frac{p_1}{p_1}p_1 + \frac{p_i}{p_2}p_2 + \ldots + \frac{p_i}{p_j}p_j + \ldots + \frac{p_i}{p_n}p_n = p_i + p_i + \ldots + p_i + \ldots + p_i = np_i ,$$

або

$$\sum_j \frac{p_i}{p_j} \cdot p_j = np_j.$$

Таким чином, добуток рядка *i* на вектор пріоритету *p* дає n разів пріоритету p_i. Помноживши всі елементи матриці порівняння *A* на вектор пріоритету *p*, отримаємо таку рівність:

$$\begin{bmatrix} p_1/p_1 & p_1/p_2 & \cdots & p_1/p_n \\ p_2/p_1 & p_2/p_2 & \cdots & p_2/p_n \\ \cdots & \cdots & \cdots & \cdots \\ p_n/p_1 & p_n/p_2 & \cdots & p_n/p_n \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{bmatrix} = n \begin{bmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{bmatrix},$$

що є рівнянням: *Ap = np*. Отже, для послідовної матриці вектор пріоритету отримують розв'язуванням даного рівняння.

Для несумісної матриці це відношення вже не є дійсним, оскільки порівняння між елементами і та j не обов'язково дається формулою: $a_{ji} = p_i / p_j$.

Тому розмірність *n* замінюється невідомим λ. Обчислення λ і *p* за таким рівнянням, як *Ap = λp*, у лінійній алгебрі називається задачею власних значень [12]. Будь-яке значення λ, що задовольняє це рівняння, називається власним значенням, а *p* - пов'язаним із ним власним вектором. Відповідно до теореми Перрона [13], позитивна матриця має унікальне позитивне власне значення. Нетривіальне власне значення називається максимальним власним значенням λ_{max}. Якщо $\lambda_{max} = n$, то матриця цілком узгоджена. В іншому випадку різниця між λ_{max} - n є мірою невідповідності.

Сааті обґрунтував підхід власних значень для несуперечливих матриць теорією збурень, яка говорить, що незначні зміни в послідовній матриці означають лише незначні зміни власного вектору та власного значення.

Для того, щоб розрахувати власний вектор, пов'язаний з максимальним власним значенням, більшість програмних пакетів, використовують метод живлення, який є ітераційним процесом:

1. Парна матриця має квадрат: $A_{n+1} = A_n A_n$.
2. Обчислюються та нормуються суми рядків. Це 1-е наближення власного вектору.
3. За допомогою матриці A_{n+1} кроки №1 та №2 повторюються.
4. Крок №3 повторюється до тих пір, поки різниця між цими сумами в двох послідовних розрахунках пріоритетів не буде меншою, ніж заданий критерій зупинки.

Розрахунок глобального пріоритету матриці наведено в табл. 17.

Таблиця – 17 (глобальні пріоритети матриці)

Альтернативи	Критерії						Глобальні пріоритети
	Стабільність	Шифрування даних	Швидкість	Сумісність клієнтів	Порти	Налаштування і конфігурація	
	Числові значення вектору пріоритету						
	0,184	*0,1844*	*0,11*	*0,07*	*0,05*	*0,03*	
OpenVPN	*0,222*	*0,2935*	*0,21*	*0,11*	*0,37*	*0,07*	*0,14451*
L2TP/IPSec	*0,222*	*0,2935*	*0,06*	*0,29*	*0,05*	*0,17*	*0,13007*
IPSec	*0,222*	*0,1322*	*0,11*	*0,29*	*0,14*	*0,36*	*0,11394*
SSTP	*0,222*	*0,2356*	*0,21*	*0,04*	*0,37*	*0,05*	*0,12815*
PPTP	*0,111*	*0,0451*	*0,41*	*0,29*	*0,08*	*0,34*	*0,10825*

Спираючись на результати розрахунку глобального пріоритету, можна стверджувати, що *OpenVPN*, зі значенням *0,14451*, являється найбільш прийнятним варіантом для цілей створення віртуальної приватної мережі.

3 Висновки

1. Вибір VPN протоколу є складним завданням, яке потребує аналізу ринків, визначення критеріїв порівняння, надання коректних пріоритетів, причому часто, в умовах, коли немає повної інформації о системі, в якій ці процеси відбуваються.
2. VPN протокол, в певної мірі, є рішенням проблеми безпечного віддаленого доступу.
3. У багатьох випадках існує необхідність приймати рішення, коли немає достатньої інформації про систему, для якої це рішення приймається. В цих випадках варто користуватися методами прийняття рішень в умовах невизначеності. Існує велика кількість таких методів, але в даній ситуації, обраними експертами був використаний метод аналізу ієрархій.
4. Даному методу притаманні наступні переваги:
   - Попарність порівнянь. Зникає необхідність тримати одночасно велику кількість альтернатив в розрахунку. Внаслідок чого отримуються більш точні результати;
   - Додатковість вихідної матриці. В реальних умовах нерідкими є випадки в котрих відбуваються зміни відношень або альтернатив. При використанні MAI це призводить лише до порівняння альтернатив, що з’явилися;
   - Наявність вербально-числової шкали;
   - Перевірка роботи експерта за допомогою відношення узгодженості;
5. Експертами, що долучалися до участі в дослідженнях визначено наступні критерії:
   - швидкість;

- шифрування даних;
- налаштування / конфігурація;
- порти;
- стабільність;
- сумісність клієнтів.

6. Як альтернативні рішення розглядалися наступні VPN протоколи: - *PPTP;* - *IPsec*; - *L2TP+IPsec*; - *SSTP та OpenVPN*. За допомогою MAI встановлено, що *OpenVPN* є найбільш прийнятним вибором при вирішенні зазначеного кола завдань.

Посилання

[1] Virtual Private Network Market Size, Share & Trends Analysis Report By Component, By Type (Site-To-Site, Remote Access, Extranet), By Deployment Mode, By End Use, By Region, And Segment Forecast, 2020 – 2027
[2] COVID-19 VPN Demand Statistics. URL: https://www.top10vpn.com/research/investigations/covid-19-vpn-demand-statistics/ (дата звернення: 15.10.2020).
[3] Lewis M. Comparing, Designing, And Deploying Vpns. Ne: Cisco Systems, 2006. 1043 c.
[4] Saaty T. Decision making with the analytic hierarchy process. // International Journal of Services Sciences. 2008. № 1 (1).
[5] Simanaviciene R. and Ustinovichius L. Sensitivity analysis for multiple criteria decision-making methods: TOPSIS and SAW. // Procedia Social and Behavioral Sciences. 2010. № 2.
[6] Kowalski, Z. Meler-Kapci M. Zielinski S. CBR methodology application in an expert system for aided design ship's engine room automation. // Expert Systems with Applications, 2005. № 29 (2).
[7] Fishburn, P. Conjoint measurement in utility theory with incomplete product sets // Journal of Mathematical Psychology, 1967. № 4(1): 104-119.
[8] Chen, Y., Okudan, G., and Riley, D. Decision support for construction method selection in concrete buildings // Prefabrication adoption and optimization. Automation in Construction. 2010. № 19(6): 665-675.
[9] Konidari P., Mavrakis D. A multi-criteria evaluation method for climate change mitigation policy instruments // Energy Policy. 2007. № 35 (12). C.6235-6257.
[10] Saaty T. The analytic hierarchy process: Planning, priority setting, resource allocation. New York: McGraw-Hill, 1980. 287 c.
[11] Oguztimur S. Why Fuzzy Analytic Hierarchy Process Approach For Transport Problems? // 51st ERSA Congress. Barcelona, 2011. C.439.
[12] Ильин В.А., Позняк Э.Г. Линейная алгебра. Москва: Наука, 1974. 292 с.
[13] Гантмахер Ф.Р. Теория матриц. Москва: Наука , 1966. 576 с.

Reviewer: Roman Oliynikov, Doctor of Sciences (Engineering), Full Prof., JSC "Institute of Information Technologies", 12 Bakulin St., Kharkiv, 61166, Ukraine.
E-mail: roliynykov@gmail.com

Received: June 2020.

Authors:
Usichenko Nikita, computer science student, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: nik.usichenko@gmail.com

Sabina Ruzudzhenk, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: ruzudzhenk.jb@gmail.com

Karina Pogorelaya, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: karina.pogorelka@gmail.com

Finding the optimal VPN-solution based on the hierarchies analysis method.

Abstract. The article considers the problems of creating a virtual private network in the modern world. A method for finding the optimal solution of a virtual private network based on the method of hierarchy analysis is proposed. The advantages and disadvantages of this method are given. Due to the availability of a wide range of software and hardware, solutions based on five current VPN protocols are considered: PPTP, IPsec, L2TP + IPsec, SSTP, and OpenVPN. Experts involved in the study introduced six criteria for determining the best VPN solution. The following criteria are speed, data encryption, settings, configuration, ports, stability, and customer compatibility. A hierarchy of decision levels for case studies is provided. The assessment of priorities on the basis of expert judgments of specialists on this topic has been performed. The consistency check has been performed to identify possible contradictions. The global priority is determined using the method of eigenvalues, which calculates not only the priorities but also the degree of inconsistency. According to the simulation results, it is emphasized that the choice of VPN protocol is a difficult task. Solving this problem requires market analysis, the definition of comparison criteria, and prioritization. Attention is drawn to the fact that all these components are carried out in conditions where there is no complete information about the system in which these processes occur. In this case, it is necessary to use decision-making methods in conditions of uncertainty. At present, there are a large number of such methods, but in this situation, experts have proposed a method of analysis of hierarchies. Under the above criteria and priorities, according to the results of the calculation, the OpenVPN protocol is the most optimal solution for creating a virtual private network.

Keywords: Method of analysis of hierarchies; Virtual Private Network; Saati method; VPN protocol.

Рецензент: Роман Олейников, д.т.н., проф., ЧАО "Институт информационных технологий", ул. Бакулина, 12, Харьков, 61166, Украина.
E-mail: roliynykov@gmail.com

Поступила: Июнь 2020.

Авторы:

Усиченко Никита, студент факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: nik.usichenko@gmail.com

Сабина Рузудженк, студентка факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: ruzudzhenk.jb@gmail.com

Карина Погорелая, студентка факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: karina.pogorelka@gmail.com

Нахождение оптимального VPN решения на основе метода анализа иерархий.

Аннотация. В работе рассмотрена проблематика по созданию виртуальной частной сети в современном мире. Предложен метод нахождения оптимального решения виртуальной частной сети на основе метода анализа иерархий. Приведены преимущества и недостатки данного метода. Из-за доступности широкого спектра программно-аппаратных средств, рассмотрены решения на основе пяти актуальных VPN протоколов PPTP, IPsec, L2TP + IPsec, SSTP и OpenVPN. Привлеченными к выполнению исследований экспертами введены шесть критериев для определения лучшего VPN решения. Введены следующие критерии: скорость, шифрование данных, настройки, конфигурация, порты, стабильность и совместимость клиентов. Предоставлена иерархия уровней принятия решений для тематического исследования. Выполнена оценка приоритетов на основании экспертных суждений профильных, в данной тематике, специалистов. Выполнена проверка согласованности для выявления возможных противоречий. Определен глобальный приоритет с помощью метода собственных значений, который вычисляет не только приоритеты, но и степень несоответствия. По результатам моделирования подчеркнуто, что выбор VPN протокола является сложной задачей. Решение этой задачи требует анализа рынка, определение критериев сравнения и предоставления приоритетов. Обращено внимание на тот факт, что, все указанные составляющие осуществляются в условиях, когда нет полной информации о системе, в которой эти процессы происходят. В таком случае, необходимо пользоваться методами принятия решений в условиях неопределенности. В настоящее время существует большое количество таких методов, но в ситуации, рассмотренной экспертами, был предложен метод анализа иерархий. В условиях приведенных критериев и предоставленных приоритетов, по результатам расчета, протокол OpenVPN является наиболее оптимальным решением для создания виртуальной частной сети.

Ключевые слова: метод анализа иерархий; виртуальная частная сеть; метод Саати; VPN протокол.

УДК 004.415.53 DOI: 10.26565/2519-2310-2020-2-02

ОГЛЯД СТАТИЧНИХ МЕТОДІВ АНАЛІЗУ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Олексій Прищепа, Олександра Доценко

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
olexiiprish@gmail.com, aleksandra.docenko99@gmail.com

Рецензент: Володимир Хома, д.т.н., проф., Опольський політехнічний Університет, Ополе, Польща
xoma@wp.pl

Поступила: Травень 2020.

Анотація: *У сучасному світі проблема втрат від дій зловмисного програмного забезпечення (в т.ч. звичайного, яке несе в собі ознаки не декларованих функцій) продовжує оставатися вкрай актуальною. Тому створення та модифікація антивірусних рішень захисту і аналізу зловмисного програмного забезпечення (ПЗ) є актуальним, та перспективним напрямком досліджень. Це обумовлено відсутністю існування єдиного, універсального способу який забезпечує стовідсоткове знаходження шкідливого коду. У роботі розглядаються склад та основні компоненти статичного аналізу. Визначено основні способи статичного аналізу, та наведені відповідні приклади майже всіх з них. Зроблено висновок, що основними перевагами статичного аналізу є те, що за допомогою використання відносно простого набору команд та інструментів, можливо виконати аналіз зловмисного програмного забезпечення, та частково зрозуміти, як він працює. Звернено увагу на той факт, що статичний аналіз не дає стовідсоткову впевненість в тому, що досліджене ПЗ є зловмисним. Враховуючи це, для забезпечення більш змістовного аналізу потрібно збирати якомога більше даних про структуру файлу, його можливі функції та ін. Аналіз файлів на можливу присутність зловмисного програмного коду, забезпечується за рахунок використання відповідних програм перегляду їх структури та складу. Більш інформативним способом є аналіз Portable Executable – формату. Він складається з аналізу різних секцій коду полів та ресурсів. Оскільки статичний аналіз не завжди надає потрібний рівень гарантій, то на етапі прийняття остаточного класифікаційного рішення (ПЗ зловмисне або ні) краще використовувати алгоритми машинного навчання. Такий підхід надає змогу обробляти великі масиви даних з більшою точністю, стосовно визначення характеру ПЗ, яке аналізується. Головною метою роботи є аналізі існуючих способів статичного аналізу зловмисного ПЗ, та огляд особливостей їх подальшого розвитку.*

Ключові слова: *аналіз; програмне забезпечення; зловмисне програмне забезпечення; статичний аналіз.*

1 Вступ

Практично кожного дня, у світі хтось створює новий вірус, «троянського коня», або експлойт [1] для будь-якої звичайної програми. Кожного дня мільйони людей та десятки компаній втрачають гроші, бази персональних даних, та іншу важливу інформацію. І при цьому кожного дня спеціалісти з інформаційної безпеки (ІБ) з різних країн, модифікують і створюють відповідні, засоби та системи захисту інформації, складають різні захисні модулі на програмні продукти, доповнюють антивірусні бази новими сигнатурами новоз'явлених зловмисних програм, блокують ресурси, що розповсюджують вірусні модулі, рекламу тощо.

В міру того, як мережеві зловмисники, стають все більш витонченими і впроваджують атаки з використанням нового, складного зловмисного ПЗ, його виявлення у системі та відповідне оперативне реагування на них, має вирішальне значення. Аналіз даних на предмет виявлення зловмисного ПЗ, став обов'язковим навиком для відповідної категорії фахівців з ІБ, в межах їх боротьби зі складним зловмисним ПЗ та цільовими атаками.

Мета даної роботи полягає в аналізі існуючих методів статичного аналізу зловмисного ПЗ, та розгляді особливостей їх подальшого розвитку. В межах роботи розглядається тільки ПЗ, що адаптовано для операційних систем (ОС) сімейства Windows.

2 Основна частина

Зловмисне програмне забезпечення (*або шкідливі програми*) - це збірний термін, що позначає шкідливу програму або код, який може завдати шкоди комп'ютерній системі [1].

В цілому, шкідливе ПЗ створюється для реалізації двох основних цілей. Перша з них зводиться до отримання вигоди від вторгнення до комп'ютера (або мережі) жертви. Напри-

клад, зловмисник пробує отримати доступ до управління комп'ютером, мережею-жертви, або викрадає чутливу (*конфіденційну*) інформацію, здійснюючи, згодом, спроби вимагання грошей. Друга група цілей зводиться до матеріальних благ [2].

Є два основних напрямку аналізу зловмисного ПЗ, це статичний та динамічний.

Статичний аналіз базується на аналізі зловмисної програми без її «запуску», наприклад, аналізу її коду за допомогою евристичних методів [3].

Динамічний аналіз, навпаки, для аналізу поведінки підозрілого ПЗ передбачає запуск та ретельне спостереження за всіма його діями [3].

Кожен з цих шляхів ділиться навпіл. Так наприклад, *статистичний аналіз* поділяється на: - *базовий* та *просунутий* статистичний аналіз. Динамічний аналіз поділяється на : - *базовий* та *просунутий динамічний аналіз*.

Статичний аналіз – це аналіз, програмного коду, структури та його функцій для визначення призначення цієї програми, без необхідності його запуску/активації [3]. Статистичний аналіз складається з наступних способів виявлення зловмисного ПЗ: 1 - багаторазове сканування антивірусом; 2 - визначення типу файлу та звірення знайденого гешу програми; 3 - пошук по строкам; 4 - перевірка інформації у *PE*-заголовку (*PE - Portable Executable*); 5 - аналіз дизасемблерного коду.

2.1 Багаторазове сканування антивірусом

Під скануванням антивірусом полягає процес послідовного аналізу потенційно зловмисного ПЗ декількома антивірусами. Сканування антивірусом це перше, що треба зробити коли з'явилася проблема. Цей спосіб не є ідеальний, бо він, в цілому, покладається на відповідні бази даних з відомими частинами коду, а також виконують зіставлення по образу (*сигнатурі*), щоб ідентифікувати підозрілий код. Проблемою є те, що автори шкідливого ПЗ можуть постійно модифікувати та змінювати код, роблячи їх цим невідомими (*т.ч. безпечними*) для антивірусних сканерів. Крім того практично всі новостворені віруси, чи віруси, які з часу свого створення поки ще не набули широкого розповсюдження, теж залишаються в певному сенсі «невидимками». Причина проста - вони ще не внесені до відповідних баз даних (*сигнатур шкідливого коду*) сканерів (*т.з. реактивність оновлень*) [4].

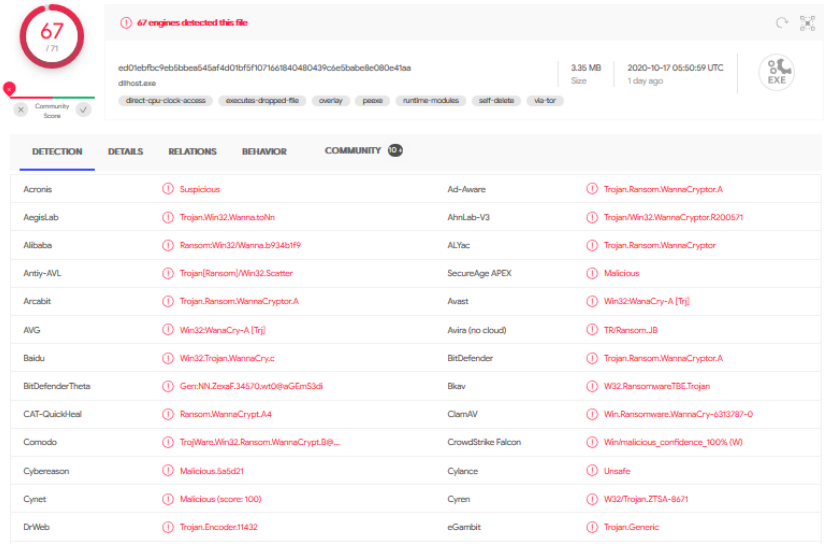


Рис. 1 – Сканування трояну *WannaCryptor*

Для сканування підозрілих файлів можна скористатися інтерактивною Інтернет службою *VirusTotal* [5]. Вона дозволяє завантажити файл, який потім сканується різними антивірусами, з видачею результатів у реальному часі. В якості прикладу на рис.1 наведено результат сканування вірусу. Бачимо, що він був перевірений 71 антивірусом, але тільки 67 змогли його ідентифікувати, як шкідливий.

На рис. 2 показаний результат сканування вірусу *AntiKiez.exe*, якій, на момент його тестування, не був ідентифікований жодним з наявних антивірусних рішень. Тобто, якщо цій вірус потрапить у комп'ютер, то не один з присутніх на ньому антивірусів не зможе зупинити його дію у системі.

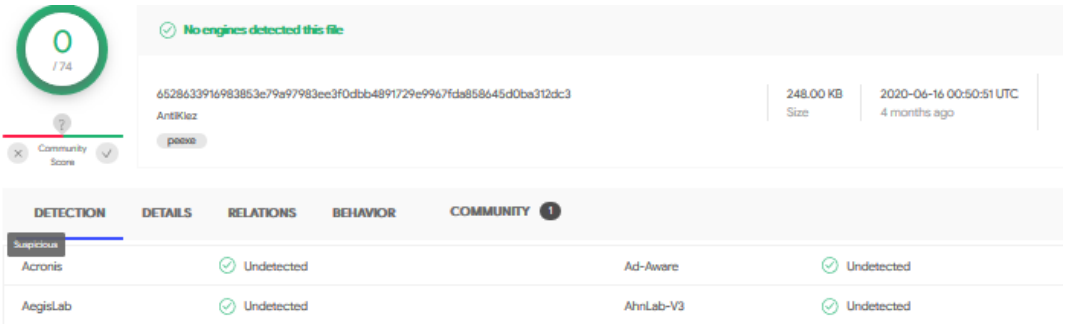


Рис. 2 – Результат сканування вірусу AntiKiez

На рис. 3 наведено результат аналізу файлу getcolor.exe. Ця програма є інструментом для знаходження потрібного кольору, та не несе в собі ніяких шкідливих дій. На логічне питання: - "Чому 3 антивіруси детектували його, як вірус?", можна припустити, що причина полягає в особливостях виконання коду програми.

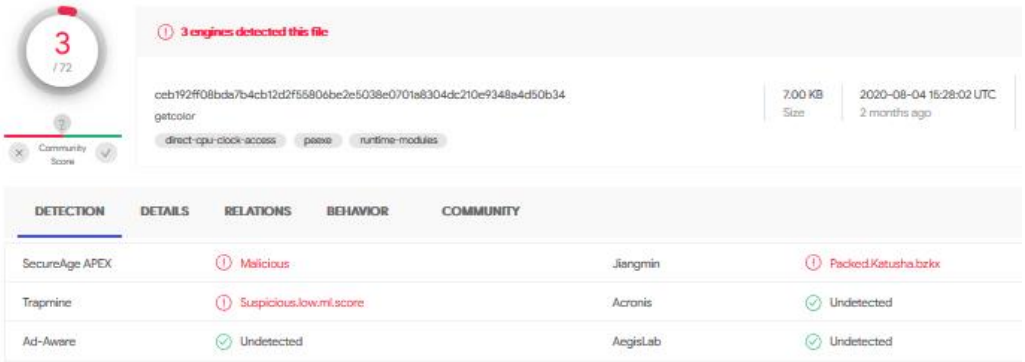


Рис. 3 – Результат сканування програми «getcolor»

2.2 Визначення типу файлу та звірення знайденого гешу програми

При аналізі типу досліджуваного файлу треба ідентифікувати на яку саме операційну систему (ОС), та яку архітектуру процесорів, перш за все націлено дане вірусне ПЗ.

Наприклад, у ОС Windows бінарні файли мають формат PE (*Portable Executable*), де це файли типу **.exe, *.dll, *.sys,* .com та ін. Але тип файлу не є гарантовано вірною інформацією, а для впевненості треба додатково перевірити цифровий підпис. Різні файли мають різні підписи, які використовуються для визначення їх типу. Перевірити цифровий підпис можливо у в hex-редакторі (рис. 4), першими двома байтами є 0x4D, 0x5A, що записуються у ASCII вигляді, як MZ [4]. Другими є сигнатура з двох PE байтів та двох нульових (0x50, 0x45, 0x00, 0x00). Третім значенням є поле Magic, воно вказує на розрядність програми, тобто, чи є файл 32-розрядним (0x010B), чи 64-розрядним (0x020B).

В цьому сенсі, в ОС Linux все декілька складніше, так як кожен файл, у якого встановлений прапор *executable,* є виконуваним.

Гешування - це поширений метод однозначної ідентифікації шкідливого ПЗ. Заражений файл «проходить» через програму хешування, в результаті чого отримується унікальний рядок (хеш), який служить ідентифікатором шкідливого ПЗ.

За цією ознакою в мережі Інтернет можна знайти цей вірус та метод боротьби з ним.

pFile	Raw Data															Value
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00	
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	i s p
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode
00000080	50	45	00	00	4C	01	02	00	BB	37	56	43	00	00	00	00	PE
00000090	00	00	00	00	E0	00	0F	01	0B	01	02	32	00	0E	00	00	
000000A0	00	0A	00	00	00	00	00	00	00	10	00	00	00	10	00	00	

Рис. 4 – Вивід початку PE файлу

Однією з найбільш поширених функцій хешування, застосовуваних аналітиками з питань інформаційної безпеки (ІБ), є *MD5* (*Message Digest 5*), хоча серія *SHA* (*Secure Hash Algorithm*) теж користується популярністю [3].

В цілому, знайдений хеш можливо використовувати в двох якостях: 1 - як ідентифікатор шкідливого ПЗ; 2 - як позначку, для пошуку у вірусних БД.

Знайти геш програми можливо з використанням відповідних рішень, т.з. геш-шифраторів. Так наприклад, на рис. 5, наведено приклад підрахунку значення гешу за алгоритмами *MD5*, *CRC32* та серії *SHA*. Отримавши відповідні значення можливо перевірити цілісність програми, або перевірити наявність його у вірусних БД.

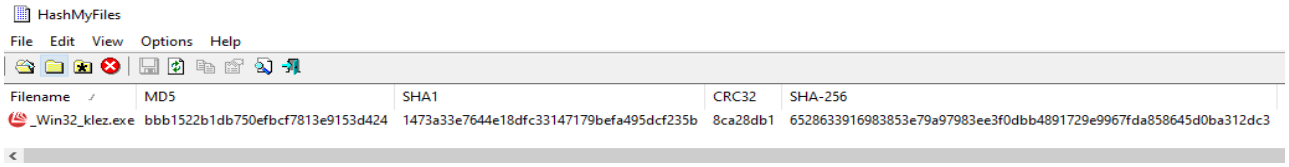


Рис. 5 – Приклад відображення гешів файлу

2.3 Пошук по строкам

Пошук по строкам може надати велику кількість корисної інформації. Наприклад, якщо програма звертається до якогось *URL*, то можливо знайти відповідну адресу, яка зберігається у вигляді строки. Строки у виконавчому файлі зберігаються у форматах *ASCII* або *Unicode*. Для здійснення пошуку, зазвичай, використовують відповідні пошукові програми. Коли така програма шукає строки в форматі *ASCII* або *Unicode*, то вона ігнорує контекст та форматування [3]. Це дозволяє аналізувати файли різних типів та знаходити потрібні строки в любому місці. Недоліком цього способу є те, що іноді знайдена послідовність може бути інтерпретована, як *строка*, хоча це можуть бути і адреса у пам'яті. Тому, в таких випадках, остаточне визначення вердикту лягає на розгляд користувача. З іншого боку, якщо користувачі отримали строки, які схожі з назвами Windows API функцій, то це вже дає додаткову інформацію про здатності програми. Як вже було зазначено вище, строки можуть містити в собі посилання на імена файлів, IP та URL-адреси, домені імена, команди для ініціювання атаки та ключі реєстру. Також, великі строки, котрі складаються з декількох і більше слів, та мають логічне формулювання речення, можуть бути повідомленням [4]. Характерним представником відповідного способу детектування зловмисного ПЗ, є програма *pestudio (*рис. 6*)*.

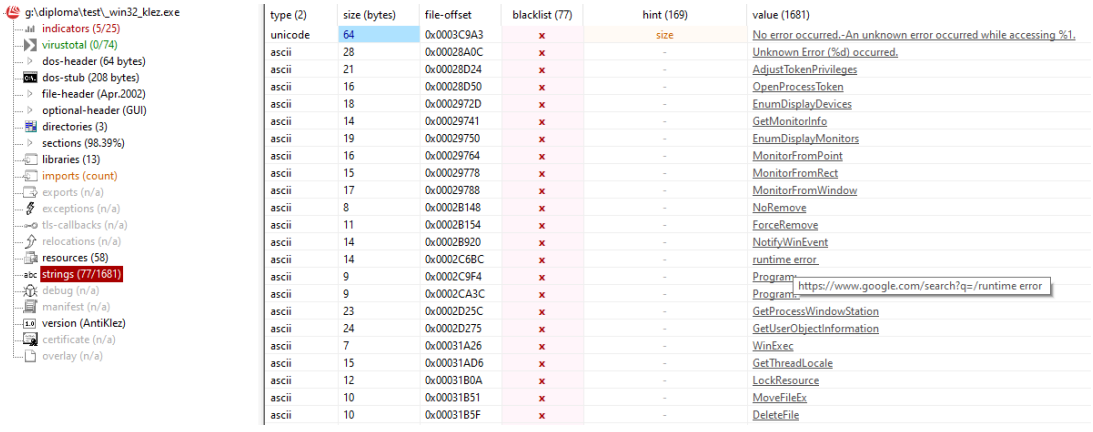


Рис. 6 – Список знайдених строк у файлі

Програма *pestudio* це багатофункціональний інструмент, який може відобразити ASCII та Unicode -строки. В наведеному на рис. 6 прикладі, можна побачити, таблицю з різними строками, де найбільший «інтерес» представляють саме рядки з поміткою «*blacklisted*».

2.4 Перевірка інформації у PE-заголовку

Переносний виконуючий формат (PE, *Portable Executable*) використовується в ОС *Windows* для виконуючих файлів, об'єктного коду та бібліотек динамічної компонування [6]. Формат PE - це структура даних, яка містить інформацію, необхідну системному завантажувачу даної ОС для управління виконавчим кодом. Практично будь-який файл для ОС Windows, в якому є виконавчий код, має формат *PE*, але іноді застарілі формати файлів все ж трапляються у шкідливих програмах.

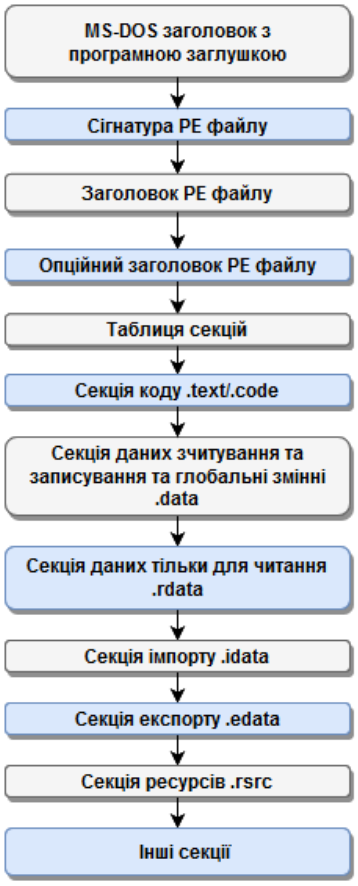


Рис. 7 – Структура образу *PE* файлу

PE-файли починаються з заголовка, який містить інформацію про код, тип додатку, необхідних бібліотечних функціях і вимогах до дисковому простору. Далі починаються секції. Секція, у вигляді коду, містить команди, які будуть виконуватися процесором, а секція, що містить дані, може являти собою різні типи даних (*зчитування та записування даних програми, таблиці імпорту та експорту, різні ресурси*).

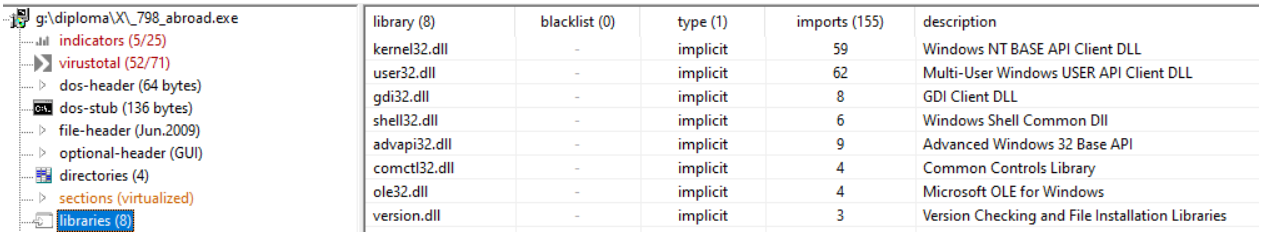
Формат *PE*-файлу організований як лінійний потік даних. Він починається з заголовка MS-DOS, програми-заглушки реального режиму і підпису *PE*-файлу. Відразу за ним слідкує *PE* заголовок і необов'язковий заголовок. Крім того, з'являються усі заголовки розділів, за якими слідують всі тіла розділів. Деякі з цих розділів не з’являються у програмі, бо не використовуються. Кінець файлу - це кілька інших областей різної інформації, включаючи: - інформацію про переміщення; - інформацію про таблиці символів; - інформацію про номер рядка і дані таблиці рядків [4]. На рис.7 наведено графічне представлення еталонної моделі *PE*-файлу.

Заголовки зберігають в собі необхідну інформацію для завантаження *PE*-файлу і першим йде DOS-заголовок.

Починати перевірку краще з кінця, бо там більше корисної інформації. Тому почнемо аналіз з файлових залежностей та імпорту. Кожна програма, яка взаємодіє з файлами, реєстром, мережею або якимось пристроєм (*гаджетом*), залежить від функцій, які експортує сама ОС. Ці функції називаються програмним інтерфейсом додатку (API), що необхідні для забезпечення взаємодії у файлах динамічно підключених бібліотек (DLL). Перевірка DLL та API-функції, які використовує шкідлива програма, може надати певні уявлення о її можливостях. Таку перевірку можна здійснити, наприклад, у програмі *pestudio*, вибравши вкладку ***libraries*** (див. Рис.8), де бачимо перелік динамічних бібліотек, які використовують у програмі. Крім того, здійснивши перехід до вкладки ***imports*** можна побачити API-функції, які імпортовані з цих бібліотек.

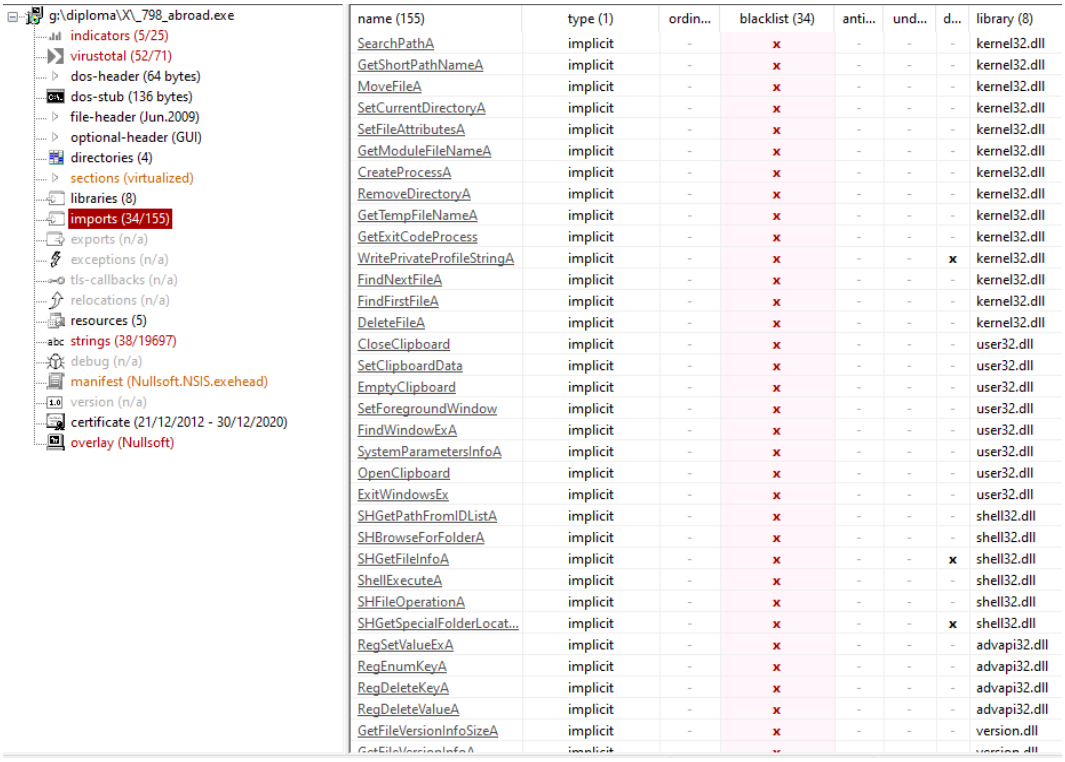
Слід підкреслити, що у програмі *pestudio* є функція виділення тих функцій, які більш ймовірно вказують на шкідливість програми, тобто на те, що програма є вірусом (див. рис. 9). Цій спосіб є доцільним в тому випадку, коли файл, що аналізується не є запакованим.

Також, на рис. 10 приведено приклад порожньої таблиці імпорту, що є підозрілим, та на це треба звертати увагу.



library (8)	blacklist (0)	type (1)	imports (155)	description
kernel32.dll	-	implicit	59	Windows NT BASE API Client DLL
user32.dll	-	implicit	62	Multi-User Windows USER API Client DLL
gdi32.dll	-	implicit	8	GDI Client DLL
shell32.dll	-	implicit	6	Windows Shell Common Dll
advapi32.dll	-	implicit	9	Advanced Windows 32 Base API
comctl32.dll	-	implicit	4	Common Controls Library
ole32.dll	-	implicit	4	Microsoft OLE for Windows
version.dll	-	implicit	3	Version Checking and File Installation Libraries

Рис. 8 – Вивід таблиці імпортованих функцій одного вірусу



name (155)	type (1)	ordin...	blacklist (34)	anti...	und...	d...	library (8)
SearchPathA	implicit	-	x	-	-	-	kernel32.dll
GetShortPathNameA	implicit	-	x	-	-	-	kernel32.dll
MoveFileA	implicit	-	x	-	-	-	kernel32.dll
SetCurrentDirectoryA	implicit	-	x	-	-	-	kernel32.dll
SetFileAttributesA	implicit	-	x	-	-	-	kernel32.dll
GetModuleFileNameA	implicit	-	x	-	-	-	kernel32.dll
CreateProcessA	implicit	-	x	-	-	-	kernel32.dll
RemoveDirectoryA	implicit	-	x	-	-	-	kernel32.dll
GetTempFileNameA	implicit	-	x	-	-	-	kernel32.dll
GetExitCodeProcess	implicit	-	x	-	-	-	kernel32.dll
WritePrivateProfileStringA	implicit	-	x	-	-	x	kernel32.dll
FindNextFileA	implicit	-	x	-	-	-	kernel32.dll
FindFirstFileA	implicit	-	x	-	-	-	kernel32.dll
DeleteFileA	implicit	-	x	-	-	-	kernel32.dll
CloseClipboard	implicit	-	x	-	-	-	user32.dll
SetClipboardData	implicit	-	x	-	-	-	user32.dll
EmptyClipboard	implicit	-	x	-	-	-	user32.dll
SetForegroundWindow	implicit	-	x	-	-	-	user32.dll
FindWindowExA	implicit	-	x	-	-	-	user32.dll
SystemParametersInfoA	implicit	-	x	-	-	-	user32.dll
OpenClipboard	implicit	-	x	-	-	-	user32.dll
ExitWindowsEx	implicit	-	x	-	-	-	user32.dll
SHGetPathFromIDListA	implicit	-	x	-	-	-	shell32.dll
SHBrowseForFolderA	implicit	-	x	-	-	-	shell32.dll
SHGetFileInfoA	implicit	-	x	-	-	x	shell32.dll
ShellExecuteA	implicit	-	x	-	-	-	shell32.dll
SHFileOperationA	implicit	-	x	-	-	-	shell32.dll
SHGetSpecialFolderLocat...	implicit	-	x	-	-	x	shell32.dll
RegSetValueExA	implicit	-	x	-	-	-	advapi32.dll
RegEnumKeyA	implicit	-	x	-	-	-	advapi32.dll
RegDeleteKeyA	implicit	-	x	-	-	-	advapi32.dll
RegDeleteValueA	implicit	-	x	-	-	-	advapi32.dll
GetFileVersionInfoSizeA	implicit	-	x	-	-	-	version.dll

Рис. 9 – Приклад списку функцій, які вважаються підозрілими

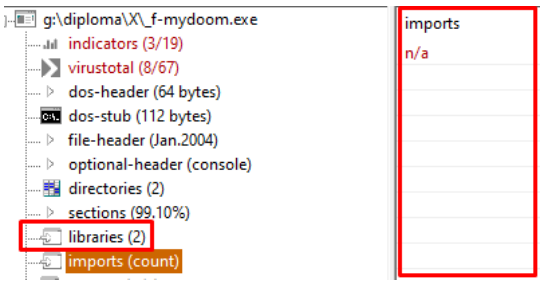


Рис. 10 – Вивід імпортованих функцій з вірусу

Наступним кроком є перевірка експорту. Майже всі випадки з експортованими функціями зустрічаються у *DLL* файлах ніж у файлах типу *EXE*. *DLL* файли здатні експортувати функції для інших програм. Перевірка експортованих функцій може надати уявлення о можливостях *DLL* (*див. список функцій на Рис. 11*). При цьому, варто підкреслити, що імена функцій експорту не завжди дають представлення о можливостях зловмисної програми, бо зловмисник може навмисно використовувати випадкові чи підставні імена, щоб більш ускладнити процес аналізу файлу.

Наступним йде аналіз таблиці секцій *PE*-файлу.

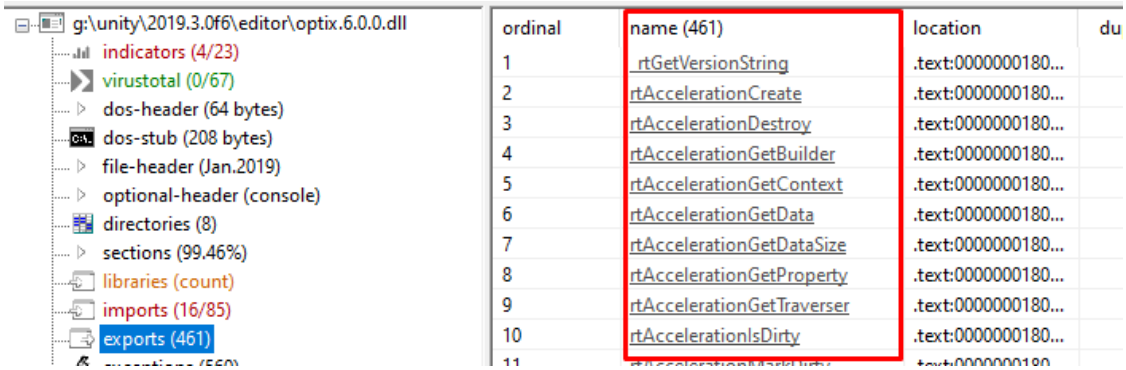


Рис. 11 – Експортовані функції

Зміст *PE*-файлу поділений на секції. Так, у прикладі, що приведено на рис. 12, є дві секції – *UPX0* та *UPX1*, що є назвою пакувальника файлів (*це все є підозрілим, тому це слід мати на увазі*). Далі можна побачити, що перша секція (*перша секція частіше це секція коду*) має характеристику ***writeble*** *(позначено, як **)*, а це значить, що вона може змінюватися. А як відомо, змінювання коду при роботі програми, є однією з характерних можливостей вірусних програм. Також за модифікування файлу відповідає мітка ***self-modifying*** *(позначено, як ***)*.

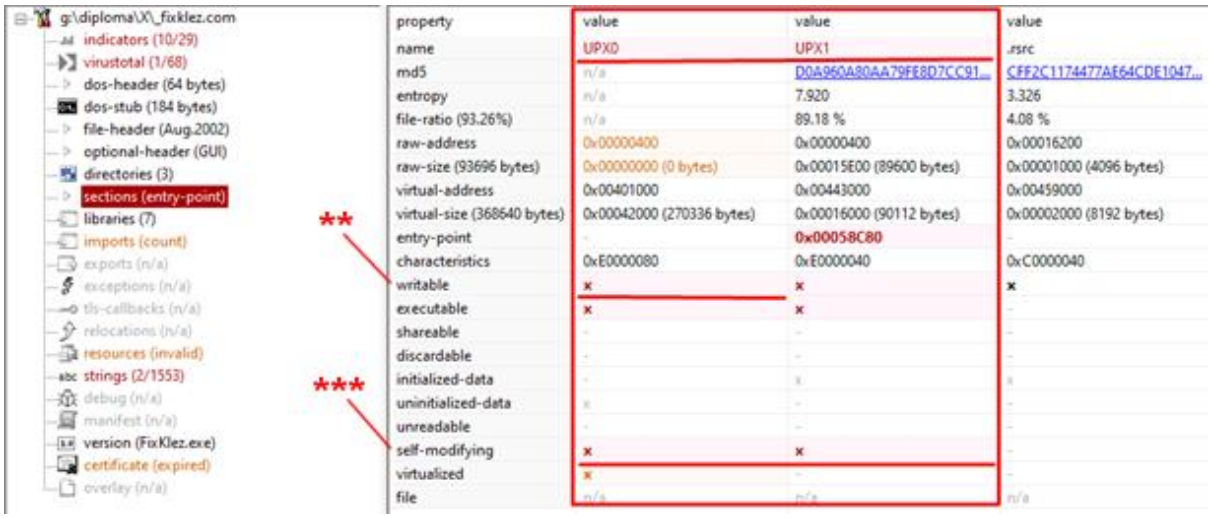


Рис. 12 – Вивід таблиці секцій

Варто зазначити, що при аналізі, також, може допомогти інформація про час компілювання програми. Вона може бути корисною, наприклад, при будуванні графіку атаки, але частіше зловмисники її змінюють. На рис. 13 показана мітка часу з датою компіляції на 1996 рік, яка, в певної мірі, може збити з пантелику, у разі якщо атака пройшла нещодавно.

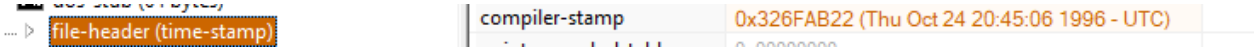


Рис. 13 – Фрагмент інтерфейсу з міткою часу

2.5 Аналіз дизасемблерного коду

Найбільш точним способом зрозуміти чи є програма шкідливою, це зробити її дизасемблювання, але це і найбільш складніший спосіб.

Дизасемблювання - вкрай копітка робота і на глибоку реконструкцію програми розміром в п'ять-десять мегабайт можуть піти багато годин. Значні трудовитрати роблять такий

спосіб аналізу надзвичайно непривабливим і, в якійсь мірі, безперспективним [3, 7]. Так, після проведення дизасемблювання підозрілого ПЗ отримаємо тисячі, чи навіть десятки тисяч строчок коду, що помітно ускладнює прямий аналіз. - Але у цьому випадку треба відштовхуватися від того, що переважна більшість вірусів, «троянів» та інших зловмисних програмних продуктів мають ряд характерних рис – своєрідних *індикаторів*, що відрізняють їх від пересічної, легітимної програми. Однак, надійність таких *індикаторів* істотно нижче і певний відсоток шкідливих програм при цьому залишиться непоміченим [7].

В цілому, кількість всіляких *індикаторів*, які прямо або побічно вказують на зараженість файлу, досить велика, але навіть вони дозволяють виявити до чотирьох з п’яти всіх існуючих вірусів [7].

Також при проведенні дизасемблювання, окрім тривалого оброблення файлу та великої кількості складно аналізованій інформації є ще одна суттєва обставина. А саме: - брати програму та відразу її дизасемблювати може не вийти, бо багато програм є «упакованими». В цьому разі можна отримати на виході безглуздий набір команд, адресів, даних та ін. Тому, спочатку треба «розібратися» з пакувальником, який може бути побудований на десятці команд, або мати і більш складний варіант, наприклад, з використанням поліморфного пакувальника, який генерує зашифровані екземпляри [3, 7].

2.6 *Рекомендації, що до покращення аналізу*

Таким чином, з урахування всього вище зазначеного, можна стверджувати, що для покращення здатності детектувати зловмисне ПЗ, треба максимально широко використовувати всі можливі шляхи аналізу.

Серед інших статичний аналіз слід використовувати в тому випадку, коли немає часу та ресурсів, чи тоді коли в наявності є тільки фрагмент файлу, коду або повідомлення.

Для більш ретельного аналізу сумнівних програм їх треба спочатку повністю розпакувати та розшифрувати. В тому випадку коли це виконавчій файл, необхідно провести його дизасемблювання. Отримавши повністю всю потрібну інформацію можна починати сегментний аналіз та інші методи знаходження шкідливих або не декларованих фрагментів коду.

Для покращення здатності прийняття рішень щодо класифікації програм, треба всіляко впроваджувати алгоритми машинного навчання. По закінченню певного етапу навчання, системи машинного інтелекту здатні автоматично аналізувати великі об’єми даних набагато швидше і точніше ніж це виконує людина.

3 Висновки

1. В роботі проведено стислий огляд відомих способів аналізу зловмисного ПЗ, та наведені основні мітки класифікації ознак.

2. Визначено, що аналіз зловмисного ПЗ ділиться на два види: статичний та динамічний. Статичний аналіз це аналіз зловмисного ПЗ без його завантаження в віртуальну пам’ять. А динамічний аналіз навпаки, реалізує аналіз поведінки вірусу у системі, та його взаємодію з мережею та системою.

3. Звернено увагу на то, що статичний аналіз є першим кроком у аналізі зловмисного ПЗ. Він дозволяє вилучити корисну інформацію, котра отримана з виконуваного файлу, та допомагає надалі у порівнянні і класифікації видів зловмисного ПЗ.

4. Підкреслено, що станом на теперішній час, базовий статичний аналіз зловмисного ПЗ складається з 5-х компонентів: - багаторазового сканування антивірусом; - визначення типу файлу та наступної звірки гешу; - пошуку по строкам; - перевірки відомостей у PE-заголовку; - аналіз дизасемблерного коду.

5. Помітною перевагою використання статичного аналізу для вирішення завдання виявлення зловмисного ПЗ є те, що за допомогою впровадження відносно простого набору команд і інструментів можливо забезпечити аналіз зловмисного ПЗ, та частково усвідомити, як він саме працює. При цьому треба пам’ятати, що статичний аналіз все-таки не дає стовідсот-

кову впевненість в тому, що проаналізована програма є зловмисною. Враховуючи це, для отримання більш об'єктивного результату аналізу, треба збирати якомога більше даних про досліджуваний файл, його структуру і можливі функції, а для прийняття рішення щодо остаточної класифікації програми (*зловмисна або ні*) потрібно використовувати алгоритми машинного навчання. Саме такий підхід надає змогу обробляти великі масиви даних з більш великою точністю, щодо результатів класифікації досліджуваних програм.

Посилання

[1] Вредоносное ПО [Електронний ресурс] / Malwarebytes Режим доступу до ресурсу: https://ru.malwarebytes.com/malware/ (дата звернення 19.05.2020) - Загл. с экрана

[2] Вредоносные программы (malvare) [Електронний ресурс] / Anti-Malware. Режим доступу до ресурсу: https://www.anti-malware.ru/threats/malware (дата звернення 19.05.2020) - Загл. с экрана

[3] Michael S. Practical Malware Analysis: The Hands – On Guide to Dissecting Malicious Software / Michael S., Andrew H.; пер. с англ. Черников С. – Санкт-Питербург : Питер, 2018. - 786 с. (Серия «Для профессионалов»).

[4] Монаппа К. А. Анализ вредоносных программ / пер. с анг. Д. А. Беликова. – М.: ДМК Пресс, 2019. – 452 с.: ил.

[5] VirusTotal Доступ до електронного ресурсу: https://www.virustotal.com/gui/

[6] Peering Inside the PE: A Tour of the Win32 Portable Executable File Format [Електронний ресурс] / Pietrek M. // Microsoft Docs. - 2010 Доступ до електронного ресурсу: https://docs.microsoft.com/en-us/previous-versions/ms809762(v=msdn.10)-?redirectedfrom=MSDN (дата звернення 11.05.2020) - Загл. с экрана

[7] Борьба с вирусами: опыт контртеррористических операций / Касперски К. // Системный администратор. – 2004. – Режим доступу до журн.: http://citforum.ru/security/virus/virii_dis/ (дата звернення 18.05.2020) - Загл. с экрана

Рецензент: Владимир Хома, д.т.н., проф., Опольский Политехнический Университет, Ополе, Польша.
E-mail: xoma@wp.pl

Поступила: Май 2020.

Авторы:

Прищепа Алексей, студент факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: olexiiprish@gmail.com

Доценко Александра, студентка факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: aleksandra.docenko99@gmail.com

Обзор статических методов анализа зловредного программного обеспечения.

Аннотация: В современном мире проблема потерь от действий вредоносных программ (или обычных, которые несут в себе признаки не декларируемых функций) продолжает быть крайне актуальной. Поэтому создание и модификация антивирусных решений защиты и анализа вредоносных программ (ПО) является актуальным, и перспективным направлением исследований. Это обусловлено отсутствием существования единого, универсального способа который обеспечивает стопроцентное нахождения вредоносного кода. В работе рассматриваются состав и основные компоненты статического анализа. Определены основные способы статического анализа, и приведены соответствующие примеры многих из них. Сделан вывод, что основными преимуществами статического анализа является то, что с помощью использования относительно простого набора команд и инструментов, можно выполнить анализ вредоносных программ, и частично понять, как он работает. Обращено внимание на тот факт, что статический анализ не дает стопроцентную уверенность в том, что исследовано ПО является злонамеренным. Учитывая это, для обеспечения более содержательного анализа нужно собирать как можно больше данных о структуре файла, его возможные функции и др. Анализ файлов на возможное присутствие вредоносных кода, обеспечивается за счет использования соответствующих программ для просмотра их структуры и состава. Более информативным способом является анализ *Portable Executable* - формата. Он состоит из анализа различных секций кода полей и ресурсов. Поскольку статический анализ не всегда предоставляет необходимый уровень гарантий, то на этапе принятия окончательного классификационного решения (злонамеренное ПО или нет) лучше использовать алгоритмы машинного обучения. Такой подход позволит обрабатывать большие массивы данных с большей точностью, по определению характера ПО, анализируется. Главной целью работы является разбор существующих способов статического анализа злонамеренного ПО, и обзор особенностей их дальнейшего развития.

Ключевые слова: Анализ; программное обеспечение; вредоносное программное обеспечение; статические методы анализа.

Reviewer: Volodymyr Khoma, Dr. of Sciences (Eng.), Full Prof., The Opole University of Technology, Opole, Poland.
E-mail: xoma@wp.pl

Received: May 2020.

Authors:

Alex Pryshchepa, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: olexiiprish@gmail.com

Aleksandra Docenko, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: aleksandra.docenko99@gmail.com

Overview of static methods of analysis malicious software.

Annotation: In today's world, the problem of losses from the actions of malicious software (or ordinary software, which has the characteristics of undeclared functions) continues to be extremely relevant. Therefore, the creation and modification of anti-virus solutions for protection and analysis of malware (software) is a relevant and promising area of research. This is due to the lack of a single, universal method that provides 100% finding malicious code. The paper considers the composition and main components of static analysis. The main methods of static analysis is identified, and relevant examples of almost all of them are given. Got concluded that the main advantages of static analysis are that by using a relatively simple set of commands and tools, it is possible to perform malware analysis and partially understand how it works. Attention is drawn to the fact that static analysis does not give 100% certainty that the investigated software is malicious. With this in mind, to provide a more meaningful analysis, you need to collect as much data as possible about the structure of the file, its possible functions, etc. Analysis of files for the possible presence of malicious code is provided through the use of appropriate programs to view their structure and composition. A more informative way is to analyze the Portable Executable format. It consists of the analysis of various sections of the code of fields and resources. Since static analysis does not always provide the required level of guarantees, it is better to use machine learning algorithms at the stage of making the final classification decision (malicious or not). This approach will make it possible to process large data sets with greater accuracy in determining the nature of the software is analyzed. The main purpose of this work is to analyze the existing methods of static malware analysis, and review the features of their further development.

Keywords: Analysis; Software, Malware; Static analysis of malware.

УДК 621.327:621.391 DOI: 10.26565/2519-2310-2020-2-03

ВЕРИФІКАЦІЯ ВІДБИТКІВ ПАЛЬЦІВ З ВИКОРИСТАННЯМ РІШЕННЯ ЗАДАЧІ КОМІВОЯЖЕРА І ДЕКОМПОЗИЦІЇ ОТОЧЕННЯ МІНУЦІЙ

Ольга Мелкозьорова, Сергій Малахов, Валерія Гайкова

Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
olha.melkozerova@karazin.ua, mailgate@meta.ua, valeriagaikova98@gmail.com

Рецензент: В'ячеслав Калашников, д.ф.-м.н., проф., Технологічний університет Монтеррея, 64849 Монтеррей, Нуево-Леон, Мексика
kalash@itesm.mx

Надійшла: Липень 2020.

Анотація. *У статті наведено приклад верифікації бази відбитків пальців методом рішення задачі комівояжера з використанням декомпозиції околиць найближчих мінуцій. Рішення цієї задачі має стійкість до лінійних, кутових деформацій, перемішуванню точок. Цей метод забезпечує правильне рішення для невеликої кількості точок, для великої кількості точок виникають перетин контурів, рішення при цьому не є оптимальним. Тому для зменшення часу обробки та розрахунку метрики наведено змінений алгоритм рішення задачі методом гілок та границь, а саме вирівнювання та виключення дуг на кожному циклі пошуку оптимального маршруту. Верифікація базується на створенні локальних структур для кожної мінуції відбитку, тому що саме локальні структури мають стійкість до деформацій. Побудова глобальних структур дуже часто не призводить до гарних показників якості, так як виникає проблема при центруванні всього зразку. Проведено повний перебір випробувань шаблонів бази даних відбитків пальців при їх верифікації цим методом. Використання декомпозиції характерних ознак забезпечує більшу стійкість при дописуванні помилкових та стиранні справжніх мінуцій. В результатах статті наведено значення парних порівнянь двох шаблонів для справжніх та помилкових випробувань. Досліджено показники помилкової відмови (FRR – false rejection rate), помилкового доступу (FAR – false acceptance rate), єдиної еквівалентної помилки (EER – equal error rate).*

Ключові слова: *відбитки пальців; мінуція; найкоротша відстань; локальні ознаки; оптимальний маршрут.*

1 Вступ

На поточний момент часу рішення задачі, що пов'язана з верифікацією відбитків пальців все ще продовжує залишатися актуальною в багатьох сферах діяльності, наприклад, таких як, криміналістика, інформаційні технології, банківська справа та ін.

У представленій роботі розглядається метрика, яка заснована на побудові *локальної структури* найменшої відстані між точками. *Локальна структура* - це така структура, система координат якої, безпосередньо пов'язана з конкретною мінуцією, а не з системою координат для відбитка в цілому. Важливо відзначити, що локальні метрики мають стійкість до всіх типів спотворень, в тому числі лінійним і кутовим спотворенням. При цьому деякі з них дозволяють виключити проблему дописування помилкових мінуцій і видалення справжніх, що є найбільш суттєвою проблемою при рішенні задачі верифікації відбитків пальців [1-3].

Для побудови структури, авторами роботи використаний базовий алгоритм вирішення задачі комівояжера [4]. Однак, як відомо, при його використанні виникають проблема перетину контурів [5] і для нього не описана можливість виключення дуг на маршруті при великій кількості ітерацій. Тому в даній статті розглянуто опис, яке дозволяє виключити зазначені труднощі і зменшити час на обробку всіх «дерев» при рішенні зазначеного вище завдання. Також, в рамках роботи, досліджені показники помилкової відмови (*FAR – false acceptance rate*) та єдиної еквівалентної помилки (*EER – equal error rate*) при повному переборі бази даних відбитків пальців.

2 Алгоритм вирішення задачі комівояжера

У даній роботі використаний алгоритм вирішення задачі комівояжера, основні етапи, якого представлені наступним чином:

1. Складання матриці взаємних відстаней, утворення матриці $n \times n$;
2. Знаходження мінімальних значень за рядками;
3. Віднімання значень, які знайдені в пункті 2, з відповідних рядків;
4. Знаходження мінімальних значень взаємних відстаней в стовпцях матриці;
5. Віднімання значень, що отримані в п. 4, з відповідних стовпців *(т.ч., принаймні, один з елементів рядка стає нулем)*;
6. Для кожного нуля знаходимо суму констант приведення, мінімальне значення по рядках і стовпцях;
7. З усіх нульових елементів вибираємо нульовий елемент з максимальною сумою констант приведення;
8. Включаємо дугу (i, j) в маршрут;
9. Виходячи з дуги, яка була включена в маршрут, виключаємо деяку дугу, шляхом введення знака ∞ *(механізм виключення дуги описаний нижче)*;
10. Викреслюємо i-й рядок, j-й стовпець з матриці;
11. Повторюємо пункти 2-10 до тих пір, поки матриця не матимемо розмір $(n-2)\times(n-2)$;
12. Дві інші дуги вибираємо з тих елементів, де не була встановлена заборона;
13. Переглядаємо отриманий маршрут на предмет наявності пересічень;
14. Видаляємо пересічення шляхом заміни двох точок на дугах *(детальний опис та приклади розглянуті нижче).*

3 Виключення дуги з маршруту

У 9-му пункті представленого вище опису алгоритму розв'язання задачі комівояжера, є т.зв. «виключення дуги з маршруту». Слід звернути увагу на те, що виключення цієї дуги є дуже важливим елементом алгоритму. Для великого числа точок, що входять в маршрут, цей аспект потребує додавання. Включення дуги (i, j) веде до утворення деякого пов'язаного шляху, який з'єднує, наприклад, точки p та q, забороняється включати в маршрут дугу (q, p). У найпростішому випадку, після вибору дуги (i, j), варто виключити дугу (j, i), тобто елемент $d_{j,i}$ слід замінити на значення ∞. Ця ситуація ілюструється нижче, на рис. 1. Так, наприклад, якщо в ході рішення отримана дуга (0,4), то замінювати на нескінченне значення, слід довжину дуги (4,0). Якщо ж в результаті рішення обрана послідовність дуг – (0,4), (2,3), а наступною є дуга (0,2), то виключати варто елемент (4,3) (рис. 2).

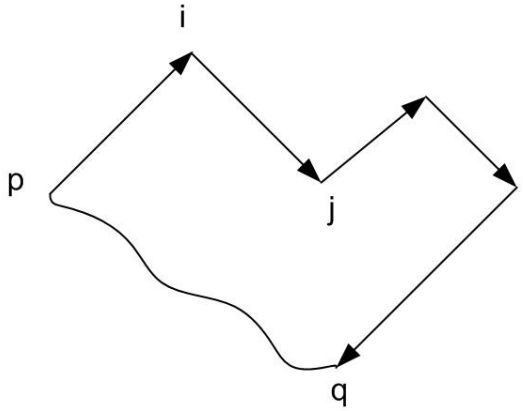


Рис. 1 – Пояснення до процедури виключення дуги

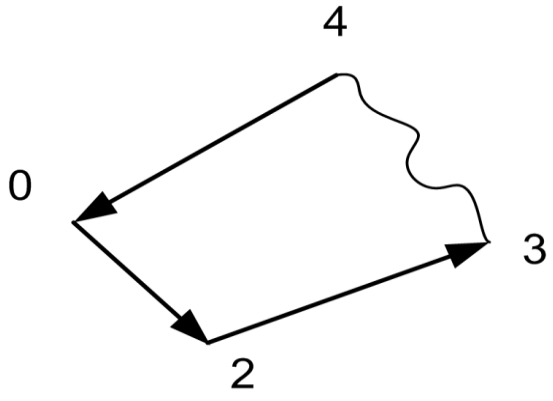


Рис. 2 – Виключення дуги (4, 3)

В основному алгоритмі описаний випадок для асиметричної матриці, яка дозволяє отримувати замкнутий контур. Однак, в разі якщо матриця симетрична і кількість точок велике, то точки, що приходять на маршрут надходять не по порядку, а довільно, і таким чином виникають складнощі з пошуком дуг на заборону. Тому, в даному випадку, слід мати на увазі, що для установки на заборону дуг, необхідно розглянути весь маршрут, який складається на даній ітерації.

Так, якщо крайні точки дуги (i, j) були в маршруті один раз, то на заборону необхідно виставити дугу (j,i). Якщо ж одна з точок була в маршруті кілька разів, то тоді, спочатку, слід видалити всі повторювані точки. Після цього слід видалити дуги на заборону, які були на попередніх ітераціях і обидві точки яких залишилися в послідовності точок без повтору. Таким чином, точки що залишилися і будуть тими точками дуги, котру слід заборонити.

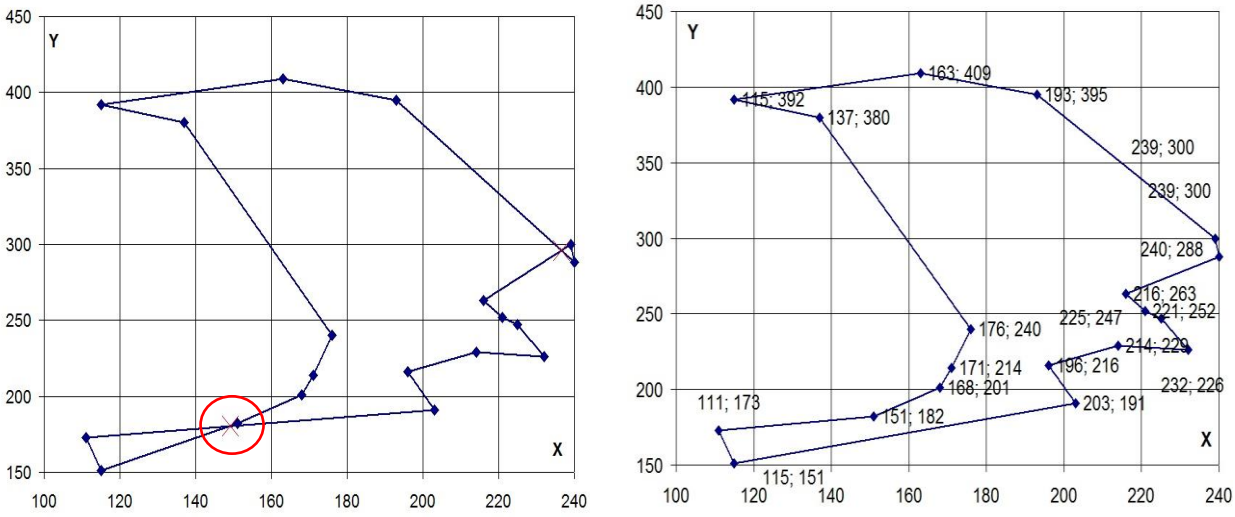
Таблиця 1 – Приклад виключення дуги з маршруту

Номер ітерації	Дуга на маршруті	Дуга для заборони
0	14, 15	15, 14
1	16, 18	18, 16
2	3, 11	11, 3
3	11, 5	3, 5
4	4, 6	6, 4
5	6, 16	4, 18

У табл. 1 наведено приклад, який дозволяє продемонструвати, логіку вибору дуги на заборону. Так, на ітерації 0, маршрут 14, 15 за правилом, описаним вище, дуга на заборону 15, 14, то ж на ітераціях 1, 2. На ітерації 3, після видалення всіх повторів з маршруту, залишаються точки 14, 15, 16, 18, 3, 5. З цього ряду видаляємо 14, 15, 16, 18 *(це дуги, які були заборонені на ітераціях 0 і 1)*, залишається дуга 3, 5, яка і є *шуканою*. На ітерації 5, після виключення всіх повторів з маршруту, залишаються точки 14, 15, 18, 3, 5, 4. З цього переліку видаляємо точки 14, 15, 3, 5, внаслідок чого залишається дуга 4, 18, котру слід заборонити.

4 Видалення пересічень контуру за базовим алгоритмом

Як вже було зазначено раніше, при вирішенні завдання комівояжера за *базовим алгоритмом*, який описаний в [1] виникають проблеми, що пов'язані з пересіченням відрізків маршруту (див. *рис. 3(а) і Табл. 2*). Так, наприклад, пересікаються відрізок з координатами (239, 300) – (216, 263) з відрізком (193,395) – (240,288), а також перетинається відрізок (203,191) – (111, 173) з відрізком (115, 151) – (151,182).



а) – рішення, що отримане за базовим алгоритмом;

б) – рішення, яке отримане після видалення пересічень.

Рис. 3 – Виключення пересічень з маршруту

Таблиця 2 – Дані для 20 точок X_i, Y_i

Точки на маршруте	X	Y	X'	Y'	X"	Y"
0	176	240	240	288	239	300
1	171	214	239	300	240	288
2	168	201	216	263	216	263
3	111	173	221	252	221	252
4	137	380	225	247	225	247
5	151	182	232	226	232	226
6	115	392	214	229	214	229
7	196	216	196	216	196	216
8	216	263	203	191	203	191
9	221	252	111	173	115	151
10	214	229	115	151	111	173
11	115	151	151	182	151	182
12	225	247	168	201	168	201
13	203	191	171	214	171	214
14	240	288	176	240	176	240
15	239	300	137	380	137	380
16	163	409	115	392	115	392
17	232	226	163	409	163	409
18	193	395	193	395	193	395
19			240	288	239	300

- $(X_i^{'}, Y_i^{'})$ - *одержане рішення;*

- $(X_i^{''}, Y_i^{''})$ - *одержане оптимальне рішення.*

Ці пересічення збільшують маршрут, а рішення при цьому не є оптимальним, тобто довжина шляху 762, а без пересічень - 739. Точки пересічення при цьому (236, 295) і (149, 180) відповідно.

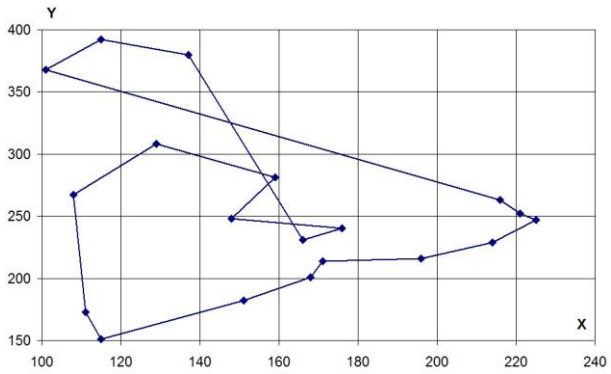
Вирішити цю проблему можна шляхом перестановки найближчих крайніх точок цих відрізків. Для цього необхідно точку з координатами (239, 300) одного відрізка, замінити на найближчу в контурі точку (240, 288). І навпаки, точку з координатами (240, 288), замінити на (239, 300). Аналогічну процедуру слід виконати з точками (111, 173) та (115, 151). Таким чином, буде сформований маршрут $X'Y'$, який є оптимальним маршрутом (*див. рис. 3(б)*).

Правило, для виключення таких пересічень працює і для більш складних контурів, дані по яких зведені в таблиці 3. Приклади контурів див. рис. 4.

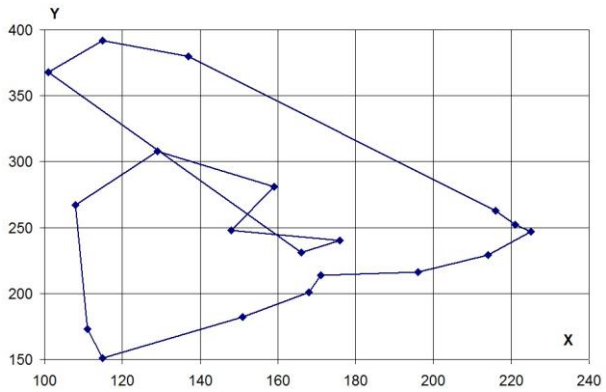
В даному випадку «вирівнювання» контуру відбувається за 5 ітерацій, але *правило*, яке описане вище, однозначно працює, і його можна з успіхом застосовувати для розв'язання задачі комівояжера.

Таблица 3– Одержане рішення $(X_i^{'}, Y_i^{'})$ і одержане оптимальне рішення $X_i^{''''''}, Y_i^{''''''}$

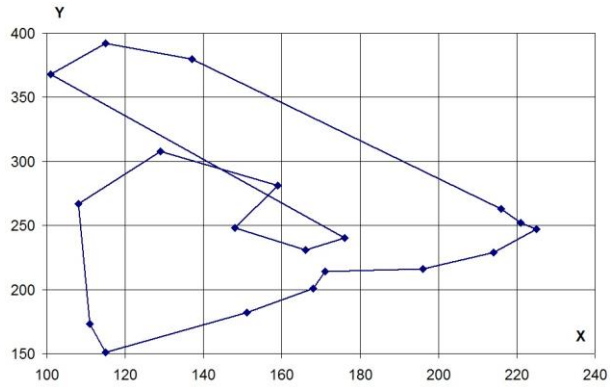
Точки на маршруті	X'	Y'	X"	Y"	X"'	Y"'	X""	Y""	X""'	Y""'	X"""	Y"""
0	111	173	111	173	111	173	111	173	111	173	111	173
1	115	151	115	151	115	151	115	151	115	151	115	151
2	151	182	151	182	151	182	151	182	151	182	151	182
3	168	201	168	201	168	201	168	201	168	201	168	201
4	171	214	171	214	171	214	171	214	171	214	171	214
5	196	216	196	216	196	216	196	216	196	216	196	216
6	214	229	214	229	214	229	214	229	214	229	214	229
7	225	247	225	247	225	247	225	247	225	247	225	247
8	221	252	221	252	221	252	221	252	221	252	221	252
9	216	263	216	263	216	263	216	263	216	263	216	263
10	101	368	137	380	137	380	137	380	137	380	137	380
11	115	392	115	392	115	392	115	392	115	392	115	392
12	137	380	101	368	101	368	101	368	101	368	101	368
13	166	231	166	231	176	240	148	248	129	308	129	308
14	176	240	176	240	166	231	166	231	166	231	159	281
15	148	248	148	248	148	248	176	240	176	240	176	240
16	159	281	159	281	159	281	159	281	159	281	166	231
17	129	308	129	308	129	308	129	308	148	248	148	248
18	108	267	108	267	108	267	108	267	108	267	108	267
19	111	173	111	173	111	173	111	173	111	173	111	173



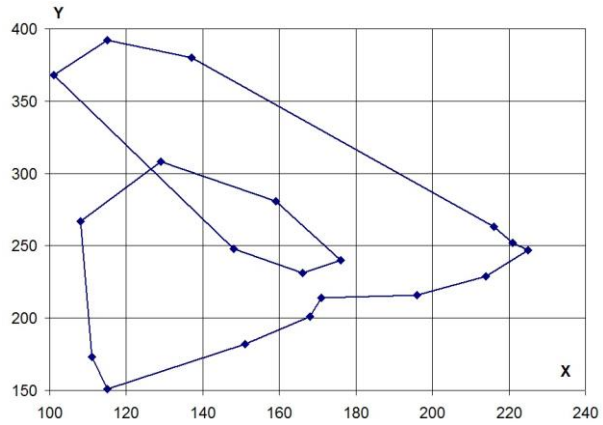
а) – опорний план $X_i^{'}, Y_i^{'}$



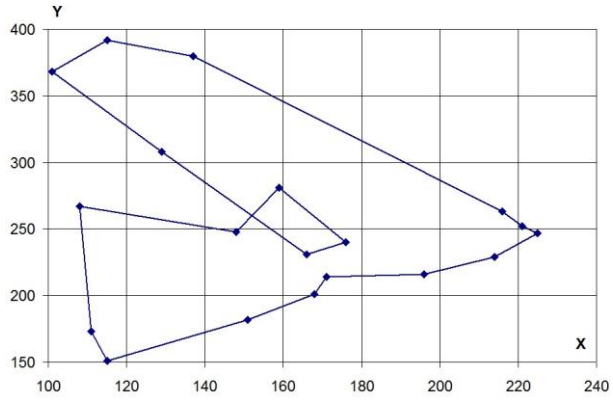
б) – перша ітерація видалення пересічень, маршрут $X_i^{''}, Y_i^{''}$



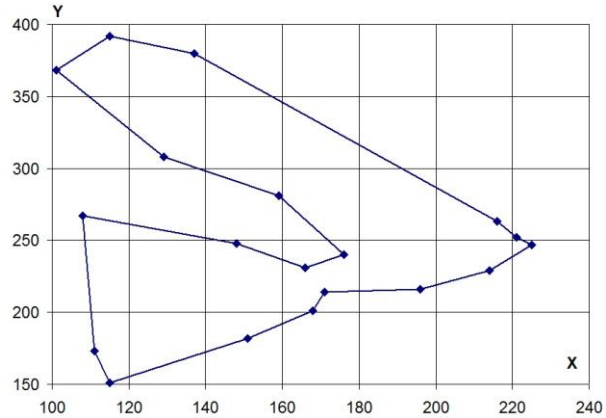
в) – друга ітерація видалення пересічень, маршрут $X_i^{'''}, Y_i^{'''}$



г) – третя ітерація видалення пересічень, маршрут $X_i^{''''}, Y_i^{''''}$



д) – четверта ітерація видалення пересічень, маршрут $X_i^{'''''}, Y_i^{'''''}$



е) – п'ята ітерація видалення пересічень, маршрут $X_i^{''''''}, Y_i^{''''''}$

Рис. 4 – Послідовність виключення пересічень маршруту *((а) →... → (е))*

5 Результати експерименту

В рамках декількох циклів моделюючих досліджень, було проведено 78400 «помилкових» випробувань та 1372 «істинних» (*це повний перебір*). За їх результатами було проведено розрахунок для десяти найближчих точок *кожної* з мінуцій. Крім того була використана декомпозиція, в результаті чого для кожної мінуцій побудовано три контури з найближчих мінуцій (див. Табл. 4), одна минуция представлена рядком з трьох чисел.

Таблиця 4 – Значення метрики

406.000	422.000	484.000
424.000	448.000	495.000
400.000	470.000	466.000
456.000	534.000	511.000
385.000	434.000	416.000
371.000	412.000	416.000
364.000	334.000	353.000
450.000	449.000	449.000
314.000	334.000	346.000
381.000	388.000	431.000
406.000	396.000	384.000

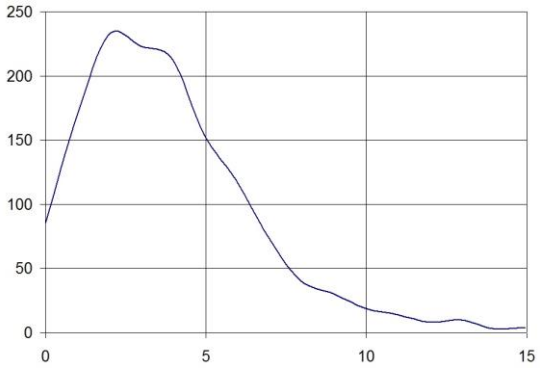


Рис. 5 – Гістограма розподілу істинних випробувань

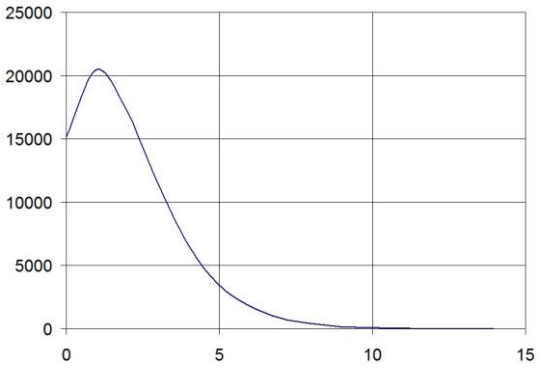


Рис. 6 – Гістограма розподілу помилкових випробувань

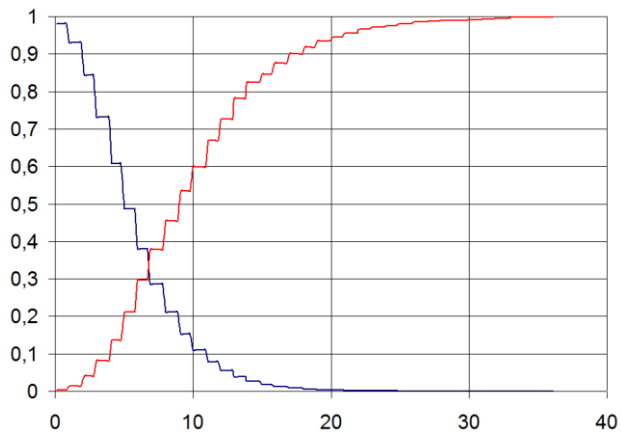


Рис. 7 – Залежності *FAR* і *FRR*

Нижче, на рис. 5-6 зображені графіки розподілів по метриці декомпозиції околиць (оточення) мінуцій.

У таблиці 5 представлені значення порівняння шаблонів для кількох «помилкових» і «дійсних» (істинних) випробувань.

На рисунках 7 та 8 наведено залежність *FAR*/*FRR* (*де, False Acceptance Rate - рівень помилкового прийняття* (*червоний*), *а False Rejection Rate - рівень помилкової відмови* (*синій*)), за якими можна визначити, що $EER \approx 33\ \%$ (*Equal Error Rate – величина еквівалентної помилки*).

Таблиця 5 – Результати порівняння шаблонів для помилкових і істинних випробувань

Значення метрики для помилкових випробувань			Значення метрики для істинних випробувань		
0_0	1_2	1.0	0_3	0_4	11.0
0_0	1_3	3.0	0_3	0_5	9.0
0_0	1_4	2.0	0_3	0_6	2.0
0_0	1_5	1.0	0_3	0_7	6.0
0_0	1_6	1.0	0_4	0_5	4.0
0_0	1_7	0.0	0_4	0_6	13.0
0_0	2_0	2.0	0_4	0_7	8.0
0_0	2_1	1.0	0_5	0_6	6.0
0_0	2_2	0.0	0_5	0_7	4.0
0_0	2_3	1.0	0_6	0_7	7.0
0_0	2_4	2.0	1_0	1_1	2.0
0_0	2_5	1.0	1_0	1_2	2.0
0_0	2_6	0.0	1_0	1_3	0.0
0_0	2_7	0.0	1_0	1_4	5.0

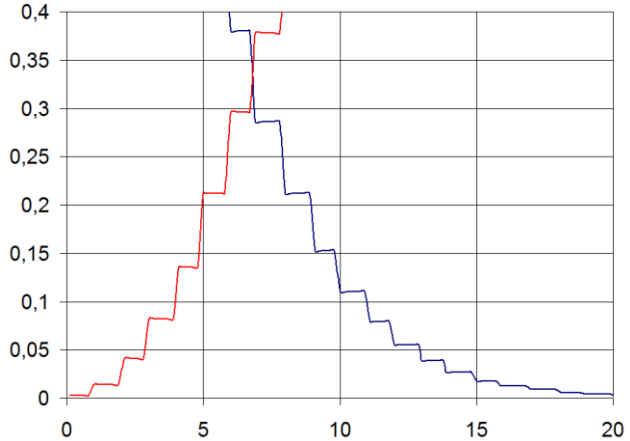


Рис. 8 – Залежності *FAR* і *FRR* (*EER =33%*)

6 Висновки

В роботі розглянута актуальна на даний момент тема – рішення задачі верифікації відбитків *методом знаходження найменшої найкоротшої відстані*, який дозволяє синтезувати локальні структури для мінуцій, які входять в шаблон.

За результатами моделювання можна стверджувати, що даний метод придатний для вирішення завдання верифікації.

До переваг даного методу слід віднести:

- відносну простоту реалізації;

- швидкість обробки бази даних. В ході експериментів час на обробку всієї бази даних становило 42 с. У порівнянні з циліндричним кодом, повний перебір тієї ж бази, займає 30 хвилин;

- повний перебір при постановці завдання верифікації становить 2 хвилини, тоді як при обробці циліндричного коду це може зайняти 24 години.

До недоліків описаного методу слід віднести низьку точність отриманих результатів (*EER = 33%*). Однак, в зв'язку з цим, справедливо буде зауважити, що на отримання оптимальних умов для тестування програми необхідні серйозні витрати часу і ресурсів.

Посилання

[1] Jin Zhe, Andrew Teoh Beng Jin, Fingerprint template protection with Minutia Vicinity Decomposition. Article, 2011. https://www.researchgate.net/publication/261431544_Fingerprint_template_protection_with_Minutia_Vicinity_Decomposition - 20.06.2020.

[2] Wajih Ullah Baig, Umar Munir, Waqas Ellahi, Adeel Ejaz, Kashif Sardar Minutia texture cylinder codes for fingerprint matching https://arxiv.org/pdf/1807.02251.pdf , Article 2018 - 20.06.2020.

[3] Melkozerova, O., Shlokin, V., Malakhov, S. Mathematical model of the biometric system of fingerprint authentication. Problems of informatization: abstracts of the reports of the seventh international conference on November 13-15, 2019, Pages. 92.

[4] Мудров В.И. Задача о коммивояжере. Издательство «Знание» Москва 1969, 61с.

[5] Melkozerova, O., Rassomakhin, S. Identification of fingers on the basis of Hamiltonian cycles of local features. the Bulletin of KNU Series "Mathematical Modeling. IT. ACS". Bulletin of V. Karazin Kharkiv National University series «Mathematical Modelling. Information Technology. Automated Control Systems». 2019. Issue 44. Pages 51–65. https://periodicals.karazin.ua/mia/article/view/15767 - 20.06.2020.

Reviewer: Vyacheslav Kalashnikov, Doctor of Sciences (Physics and Mathematics), Full Prof., Department of Systems and Industrial Engineering, Campus Monterrey, Monterrey, Mexico.
E-mail: kalash@itesm.mx

Received on July 2020.

Authors:
Olha Melkozerova, Ph.D., Senior Lecturer, Department of Security of Information Systems and Technologies, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: olha.melkozerova@karazin.ua

Serhii Malakhov, Ph.D., Senior Researcher, Associate Professor, Department of Security of Information Systems and Technologies, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: mailgate@meta.ua

Valeriia Haikova, CSD Student, V.N. Karazin Kharkiv National University, Ukraine.
E-mail: valeriagaikova98@gmail.com

Fingerprint verification using the traveling salesman problem solution and decomposition of the vicinity of the minutiae.

Abstract. The article presents an example of verification of the fingerprint database by the method of solving the problem of a salesman using the decomposition of the neighborhood of the nearest minutes. The solution of this problem is resistant to linear, angular deformations, mixing of points. This method provides the correct solution for a small number of points, for a large number of points there is a cross section of the contours, the solution is not optimal. Therefore, to reduce the processing time and calculate the metric, a modified algorithm for solving the problem by the method of branches and boundaries, namely the alignment and exclusion of arcs on each cycle of the optimal route. Verification is based on the creation of local structures for each minute of the imprint, because it is the local structures that are resistant to deformation. Building global structures very often does not lead to good quality indicators, as there is a problem with the centering of the entire sample. A complete list of tests of fingerprint database templates during their verification by this method has been carried out. The use of decomposition of characteristic features provides greater stability when adding false and erasing true minutes. The results of the article show the values of pairwise comparisons of two templates for true and false tests. The indicators of false rejection rate (FRR), false access rate (FAR), single equivalent error rate (EER) were studied.

Keywords: Fingerprints; Minutia; The Shortest Distance; Local Signs; The Optimal Route.

Рецензент: Вячеслав Калашников, д.ф.-м.н., проф., Технологический университет Монтеррея, Монтеррей, Мексика.
E-mail: kalash@itesm.mx

Поступила: Июль 2020.

Авторы:

Ольга Мелкозерова, к.т.н., доцент кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: olha.melkozerova@karazin.ua

Сергей Малахов, к.т.н., с.н.с., доцент кафедры Безопасности информационных систем и технологий, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: mailgate@meta.ua

Валерия Гайкова, студентка факультета компьютерных наук, Харьковский национальный университет имени В. Н. Каразина, Харьков, Украина.
E-mail: valeriagaikova98@gmail.com

Верификация отпечатков пальцев с использованием решения задачи коммивояжера и декомпозиции окрестностей минуций.

Аннотация: В статье проведена верификация базы отпечатков методом решения задачи коммивояжера с применением декомпозиции окрестностей близлежащих минуций. Решение данной задачи обладает устойчивостью к линейным, угловым искажениям, перемешиванию точек. Данный метод обеспечивает правильное решение для небольшого числа точек, при большом количестве точек возникают пересечения контуров, решение при этом не является оптимальным. Поэтому для уменьшения времени обработки данных и расчета метрики проведено изменение алгоритма решения задачи методом ветвей и границ, а именно выравнивание контура и исключение дуг на каждом цикле поиска оптимального маршрута. Верификация основывается на создании локальных структур для каждой минуции отпечатка, так как именно локальные структуры обладают устойчивостью к искажениям всякого рода. Построение глобальных структур зачастую не приводит к хорошим показателям точности, так как возникает проблема при центрировании всего образца. Проведен полный перебор испытаний образцов базы данных отпечатков пальцев при их верификации этим методом. Применение декомпозиции характерных признаков обеспечивает большую устойчивость при дописывании ложных и стирании истинных минуций. В результатах статьи приведены значения парных сравнений двух шаблонов для истинных и ложных испытаний. Исследованы показатели ложного отказа (*FRR – false rejection rate*), ложного доступа (*FAR – false acceptance rate*), единой эквивалентной ошибки (*EER – equal error rate*).

Ключевые слова: отпечатки пальцев; минуции; кратчайшее расстояние; локальные признаки; оптимальный маршрут.

УДК 004.056.5 DOI: 10.26565/2519-2310-2020-2-04

СПОСІБ КРИПТОЛОГІЧНИХ ПЕРЕТВОРЕНЬ ДАНИХ

Михайло Сукнов[1], Ігор Громико[2], Євгеній Перчик

[1] Харківський національний університет радіоелектроніки. пр. Науки, 14, Харків, 61166, Україна
[2] Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, 61022, Україна
mykhailo.suknov@nure.ua, i.gromyko@karazin.ua

Рецензент: Володимир Максимович, д.т.н., проф., Національний університет «Львівська політехніка», м. Львів, 79013, Україна.
vmax@polynet.lviv.ua

Поступила: Липень 2020.

Анотація: *Протидія квантовому комп'ютеру в процесі неповноважного надшвидкісного розшифрування повідомлень - технічно здійсненна. Власник інформації повинен протиставити комп'ютеру конкурента завдання, вирішення яких потребує нескінченне число операцій при її дешифруванні. Наприклад, - залежність функцій від нескінченного числа інформативних ознак. Власник шифрує шляхом інтегрування функцій, одержувач дешифрує рішенням інтегральних рівнянь. В цьому разі превалює не дискретний, а аналоговий підхід. Базис для реалізації такого підходу створили польські вчені. Математик Стефан Банах (Stefan Banach, 1892-1945), який створив сучасний функціональний аналіз і Маріан Мазур (Marian Mazur, 1909-1983) автор «Якісної теорії інформації». Їх теорія була створена на противагу «Кількісної теорії інформації». Фахівці з криптології, які все життя присвятили вдосконаленню «дискретної» теорії і виявилися наближеними до влади (і фінансів), намагаються не згадувати, що Клод Шеннон у своїй фундаментальній роботі «Теорія зв'язку в секретних системах» неодноразово підкреслив дискретну спрямованість своїх розробок, попередивши майбутніх дослідників про специфічну обмеженість своєї роботи, яка адаптована до теорії зв'язку. Забуваючи про необмежені швидкості і обсяги пам'яті квантових комп'ютерів, ортодокси говорять про надмірність, та інші суто технічні моменти,застосовуючи проти своїх опонентів всі можливі важелі протидії. Прогрес науки зупинити неможливо. Дослідження показали реальність створення подібних криптографічних систем постквантового рівня.*

Ключові слова*: криптологія; квантовий комп'ютер; технології автоматизованого шифрування - дешифрування.*

1 Вступ

Спосіб, що розглядається в межах даної роботи, відноситься до області криптологічних способів та пристроїв для шифрування, та дешифрування повідомлень (*інформації*).

Відомо, що існує спосіб аналітичних криптологічних перетворень за допомогою матриць [1]. У цьому способі для реалізації шифрування даних цілочислова квадратна матриця множиться на вектор, елементи якого є номерами букв алфавіту. Відповідно при дешифруванні проводиться аналогічна процедура з використанням зворотної матриці, отриманої відомим способом. Недоліком даного способу є низький порядок матриць, що обумовлений обмеженістю обчислювальної потужності використовуваних технічних засобів [2], а також – неможливість шифрування неперервних функціональних залежностей.

У вищезазначеному способі криптологічні перетворення даних здійснюються шляхом дії оператора (у формі цілочисельної квадратної матриці) на вектор-строчку частини повідомлення з можливістю наступного дешифрування, шляхом реалізації аналогічної процедури з використанням зворотної матриці.

З огляду на все вищесказане, автори даної роботи пропонують розглянути новий спосіб криптологічних перетворень даних, які мають форму неперервних однозначних функціональних залежностей. Згідно попередньому задуму, шифрування повинно здійснюватися шляхом впливу на безперервну функцію повідомлення відповідним оператором, що представляє собою інтеграл з замкненим ядром, в який перетворюється матриця, коли її порядок спрямовується до нескінченності. Це призводить до інтегрального рівняння Фредгольма першого роду, для якого існує ефективний алгоритм чисельної реалізації. При цьому повідомлення являє собою і/або – аналітичну функціональну залежність, та/або – запис мовного сигналу як безперервного графіка, та/або – повідомлення, що являє собою фрагмент відео-зображення,

та/або – повідомлення, що являє собою букви алфавіту і інші символи, яким поставлені у відповідність неперервні однозначні функції, та/або – повідомлення, що являє собою таблицю, дискретні значення якої попередньо підлягають апроксимації, перетворюючись у неперервну однозначну аналітичну залежність.

Крім того, додатково, можна підбирати інтегральний оператор таким чином, щоб зворотний йому оператор, був раціональний в сенсі ефективності процедури дешифрування, а саме – вирішення відповідного інтегрального рівняння у аналітичному вигляді, прийнятному для процедури обчислювання.

2 Основна частина

Сутність вирішуваного завдання полягає в тому, що при представленні даних у формі алфавіту відсутня можливість передачі даних у інших неперервних формах представлення, наприклад, графіків функцій, відео зображення, таблиць, запису мовного сигналу.

В даному випадку під даними розуміються:

- відео-зображення по типу кардіограм, гістограм і т.п., а також елементи, які в комплексі складають графіки більш складного виду (*неоднозначно залежать від змінної*);
- таблиця чисел, яка за допомогою апроксимації (інтерполяції) може бути представлена у вигляді аналітичного виразу або графіка;
- текст, що містить букви, слова, числа, формули та інші позначення, які можуть бути представлені у вигляді графіка, наприклад, амплітуди акустичного тиску запису мовного сигналу;
- аналогічний попередньому текст, в якому кожному елементарному позначенню (*букви, числа, пробілу і т.і.*), ставиться у відповідність безперервна функція змінної, наприклад,

$$\psi_s(x) = c_s \sin \omega x, \tag{1}$$

де c_s – амплітуда, довільно встановлена різною для кожного $s = 1,2\ldots$.

Представлення даних у визначених формах забезпечує подальше їх аналітичне перетворення за допомогою математичного апарату безперервного аналізу, а саме, диференційного та інтегрального числення.

Згідно з пропозицією, шифрування здійснюється шляхом впливу на безперервну функцію повідомлення оператором, що представляє собою інтеграл з замкненим ядром, в який перетворюється матриця, коли її порядок спрямовується до нескінченності, що призводить до інтегрального рівняння Фредгольма 1-го роду, для якого існує ефективний алгоритм чисельної реалізації.

В якості прототипу автори пропонують використовувати відомий метод (спосіб) аналітичного шифрування на основі матричного аналізу [1, с. 20-22]. В такому випадку слова розбиваються на блоки з *n* букв (*у наведеному прикладі* $n = 3$). Кожній букві ставиться у відповідність її номер в алфавіті, тобто, ціле число. Таким чином, формується вектор-рядок:

$$\psi = \left\| \psi_j \right\|, \; j = 1, 2, 3. \tag{2}$$

Шифрування здійснюється шляхом множення на цей вектор ціло-чисельної матриці того ж порядку

$$A_n = \left\| a_{ij} \right\|, \; i, \; j = 1, 2, 3, \tag{3}$$

в результаті чого маємо

$$A_n \psi = \sum_{j=1}^{n=3} a_{ij} \psi_j = f_i, \; i = 1, 2, 3, \tag{4}$$

де f_i – також цілі числа. Знаючи ці числа і матрицю (3), «законний» одержувач може без похибки дешифрувати блок з чисел (2) за допомогою перемноження зворотної матриці на вектор-рядок f :

$$\psi = A_n^{-1} f,\ f = \left\| f_i \right\|. \tag{5}$$

Проте зі збільшенням n в (2) – (5), тривалість обчислень, пов'язаних з перемноженням елементів матриць різко зростає. Так, рішення системи лінійних алгебраїчних рівнянь за правилом Крамера з $n = 20$, де використовуються аналогічні перемноження, вимагає зовсім нереалістичних витрат машинного часу [2, с. 43-44]. Дійсно, фігурує порядок 10^7 років. Тому, з цієї точки зору, варіант $n \to \infty$ може здатися ірраціональним.

І, тим не менш, нехай у нас є вектор-рядок з великої кількості n, чисел, що розташовуються на відрізку $x \in [0, 1]$ з інтервалом $\Delta\xi$. Використовуючи в (4) позначення

$$a_{ij} = k_{ij} \Delta\xi,\ i,\ j = 1, 2, ..., n$$

(*де* k_{ij}, *на відміну від елементів матриці* a_{ij}, *не обов'язково цілі числа*), маємо наступне

$$A_n \psi = \sum_{j=1}^{n} k_{ij} \psi_j \Delta\xi = f_i,\ i = 1, 2, ..., n. \tag{6}$$

В разі $n \to \infty$, вектор Ψ та матриця A_n переходять в безперервні функції:

$$A\psi_n \to (A\psi)(x);\ k_{ij} \to k(x, \xi);\ \psi_j \to \psi(\xi);\ \Delta\xi \to d\xi;\ f_i \to f(x),$$

внаслідок чого система лінійних алгебраїчних рівнянь (6) перетворюється на інтегральне рівняння Фредгольма першого роду:

$$(A\psi)(x) = \int_0^1 k(x, \xi) \psi(\xi) d\xi = f(x),\ x \in [0, 1]. \tag{7}$$

Отже, якщо в (4) для перетворення початкового вектора Ψ до, т. з. «невпізнаного виду» f, ми мали $3 \times 3 = 9$ цілих чисел, то ядро $k(x, \xi)$, для реалізації тієї ж мети, має у своєму розпорядженні набір нескінченної кількості дійсних чисел. І, тим не менш, рівняння (7) є абсолютно особливий об'єкт, зокрема, з тієї причини, що його ядро не можна вибирати довільним. Справді, якщо, наприклад:

$$k(x, \xi) = k_0(\xi);\ k(x, \xi) = x k_1(\xi),$$

то, інтегруючи за формулою (7), отримуємо $f = b_0$ та $f = b_1 x$, де b_0, b_1 – відповідні константи. - Інакше кажучи, відбувається «знищення» інформації про функцію $\psi(x)$.

Щоб уникнути цього, а також для забезпечення єдності розв'язку рівняння (7), оскільки вихідний текст, який визначається функцією, однозначний, ядро має бути замкнутим. Останнє означає, що

$$\int_0^1 k(x, \xi) \varphi(\xi) d\xi = 0,\ x \in [0, 1]$$

лише в тому випадку, коли функція $\varphi(x) = 0$.

Множина відповідних в даному контексті ядер є практично необмеженою; їх можна, за загальним правилом, конструювати. Як приклад наведемо ядро

$$k(x, \xi) = \begin{cases} (1 - x)\xi,\ 0 \le \xi \le x; \\ x(1 - \xi),\ x \le \xi \le 1; \end{cases}$$

за допомогою нього, наприклад, для елементів (1) шифрування за формулою (7) проводиться в аналітичному вигляді, оскільки відповідні інтеграли є табличними. Справді, тут присутні інтеграли виду

$$\int_0^x \xi \sin \omega\xi d\xi = \frac{\sin \omega\xi}{\omega^2} - \frac{\xi \cos \omega\xi}{\omega} \Bigg|_0^x = \frac{\sin \omega x}{\omega^2} - \frac{x \cos \omega x}{\omega}.$$

Слід зауважити, якщо інтегральне рівняння (7) має кінцеве число різних рішень (*інакше кажучи, не є замкнутим*), то принципових ускладнень внаслідок цього також не виникає. Законному (*тобто легітимному*) одержувачу потрібно лише, додатково передати набір відповідних коефіцієнтів при власних функціях ядра $k(x, \xi)$.

Звертаючи увагу на особливості властивостей інтегрального рівняння (7), можна відзначити, що не знаючи структури оператора A (*в першу чергу, мається на увазі ядро* $k(x, \xi)$), визначення функції $\psi(x)$ є нездійсненним, навіть теоретично.

Традиційно рівняння такого типу розглядається в постановках теорії некоректних задач. При цьому рішення фактично знаходиться засобами обчислювального експерименту, що не придатне для автоматизованого алгоритму $f \to \Psi$. Іншими словами, образно висловлюючись: - біля кожного бухгалтера не можна посадити математика високої кваліфікації, діяльність якого, до того ж, може бути досить тривалою.

Спеціального виду моделювання обробки інформації процедурою інтегрування дозволяє позбутися від згаданого фактора некоректності, внаслідок чого алгоритм $f \to \Psi$ стає: по-перше, обчислювально стійким, а по-друге, – легко піддається формалізації (*тобто дешифруванню в автоматичному режимі*) [3].

3 Математична реалізація

У загальних рисах суть пропозиції, щодо вирішення відповідного інтегрального рівняння у аналітичному вигляді, прийнятному для процедури його обчислювання, розглянута нижче.

Відомо,що інтегральних операторів, за допомогою яких може здійснюватися шифрування $A\psi = f$, а потім дешифрування $\psi = A^{-1} f$, в аналітичному вигляді (*що дуже важливо*), є велика кількість. Наведемо відповідні приклади.

1. Шифрування:

$$(A\psi)(x) = \int_0^x \frac{\psi(\xi)\,d\xi}{x+\xi} = f(x) = \sum_{n=0}^{N} a_n x^n;$$

дешифрування, після розкладання $f(x)$ в степеневий ряд, інакше кажучи, формула обернення [4]:

$$\psi(x) = \sum_{n=0}^{N} \frac{a_n}{b_n} x^n,\ b_n = (-1)^n \left[\ln 2 + \sum_{m=1}^{n} \frac{(-1)^m}{m} \right].$$

2. Шифрування:

$(A\psi)(x) = \psi(x) - \int_\alpha^x g(x)h(\xi)\psi(\xi)\,d\xi = f(x),$ де як $g(x)$, так і $h(x)$ – довільні функції.

Дешифрування [4, с. 162]: $\psi(x) = f(x) + \int_\alpha^x R(x, \xi)\psi(\xi)d\xi,$

де $R(x, \xi) = g(x)h(\xi)\exp\left[\int_\xi^x g(\xi)h(\xi)\,d\xi\right].$

3. Шифрування:

$$(A\psi)(x) = \int_{-\infty}^{\infty} \frac{\psi(\xi)\,d\xi}{\xi - x} = f(x);$$

дешифрування [4, с. 193]:

$$\psi(x) = \frac{1}{\pi^2} \int_{-\infty}^{\infty} \frac{f(\xi)\,d\xi}{\xi - x},$$

де сингулярні інтеграли тлумачаться у сенсі головного значення по Коші.

При цьому слід мати на увазі наступні особливості:

- шифрування може проводитися шляхом послідовного застосування декількох інтегральних операторів (*аналогічно застосовуються формули обернення*);
- передбачається поділ масиву інформації на частини, з варіюванням операторів інтегрування, а також їх параметрів і функцій за спеціальною програмою;
- ця програма, включає реалізацію формул звернення, або ж алгоритми розв'язання інтегральних рівнянь, що функціонують у легітимного одержувача інформації в автоматичному режимі;
- власне процеси передачі повідомлень не розглядаються, в даному випадку, акцент робиться на тому, що вони хоч і розміщені в мережі, однак не є доступні для дешифрування без відповідного програмного забезпечення;
- в методологічному аспекті досягається спряженість з криптологією в її традиційному форматі, оскільки напрацьований апарат перестановок, підстановок та ін. може використовуватися і відносно сукупності елементів (1).

Розглянемо конкретний приклад шифрування і дешифрування в аналітичному вигляді для інтеграла [4]:

$$\int_0^x (ax - a\xi + c)\psi(\xi)\,d\xi = f(\xi), \tag{8}$$

де «*a*» та «*c*» – довільні константи. При цьому, законному одержувачу і, можливо, іншої не уповноваженої особі (*порушнику*) доступний зашифрований сигнал $f(x)$. Але, на відміну від порушника, технічні засоби легітимного одержувача мають в своєму складі програмну реалізацію вирішення інтегрального рівняння Вольтерра (8). Крім того, є велика вірогідність того, що сторонній одержувач, скоріш за все, не буде мати до неї доступу.

Відповідна програма автоматично здійснює перетворення $f(x) \to \Psi(x)$, тобто, робить дешифрування (*відновлення*) вихідної функції $\Psi(x)$. У програму також закладений алгоритм, згідно з яким можуть варіюватися параметри «*a*» і «*c*» на інтервалі [*0, x*].

Рішення інтегрального рівняння (8) має вигляд:

$$\psi(x) = \frac{1}{c}\frac{d}{dx}\left[\exp\left(-\frac{a}{c}x\right)\int_0^x \exp\left(\frac{a}{c}\xi\right)\frac{d}{d\xi} f(\xi)\,d\xi\right]\cdot \tag{9}$$

Нехай у (8) функція $\psi(x) = \sin\omega x$. Зокрема, це може бути буква, або символ (1), або ж член ряду Фур'є, що представляє функцію складного виду. Тоді зашифрована (*шляхом інтегрування (8)*) функція має вигляд

$$f(x) = \int_0^x (ax - a\xi + c)\sin\omega\xi\,d\xi = -\frac{ax}{\omega}\cos\omega\xi - a\left(\frac{\sin\omega\xi}{\omega^2} - \frac{\xi\cos\omega\xi}{\omega}\right) -$$

$$-\frac{c}{\omega}\cos\omega\xi\Big|_0^x = \frac{1}{\omega}(ax + c) - \frac{c}{\omega}\cos\omega x - \frac{a}{\omega^2}\sin\omega x, \tag{10}$$

відповідно в (9)

$$\frac{d}{d\xi} f(\xi) = \frac{a}{\omega} + c\sin\omega\xi - \frac{a}{\omega}\cos\omega\xi\cdot$$

Далі отримуємо в (9) інтеграл

$$\int_0^x \exp\left(\frac{a}{c}\xi\right)\left(\frac{a}{\omega} + c\sin\omega\xi - \frac{a}{\omega}\cos\omega\xi\right)d\xi = \frac{c}{\omega}e^{\frac{a}{c}\xi} +$$

$$+\frac{ce^{\frac{a}{c}\xi}}{\left(\frac{a}{c}\right)^2 + \omega^2}\left(\frac{a}{c}\sin\omega\xi - \omega\cos\omega\xi\right) - \frac{\frac{a}{\omega}e^{\frac{a}{c}\xi}}{\left(\frac{a}{c}\right)^2 + \omega^2}\left(\frac{a}{c}\cos\omega\xi +\right.$$

$$+\omega\sin\omega\xi)\Big|_0^x=\frac{c}{\omega}e^{\frac{a}{c}x}+\frac{ce^{\frac{a}{c}x}}{\left(\frac{a}{c}\right)^2+\omega^2}\left(\frac{a}{c}\sin\omega x-\omega\cos\omega x\right)-$$

$$-\frac{\frac{a}{\omega}e^{\frac{a}{c}x}}{\left(\frac{a}{c}\right)^2+\omega^2}\left(\frac{a}{c}\cos\omega x+\omega\sin\omega x\right)-\frac{c}{\omega}-\frac{c(-\omega)}{\left(\frac{a}{c}\right)^2+\omega^2}+\frac{\frac{a}{\omega}\frac{a}{c}}{\left(\frac{a}{c}\right)^2+\omega^2}. \qquad (11)$$

В цьому виразі його три останні складові в результаті перетворень дорівнюють нулю, тобто:

$$-\frac{c}{\omega}-\frac{c(-\omega)}{\left(\frac{a}{c}\right)^2+\omega^2}+\frac{\frac{a}{\omega}\frac{a}{c}}{\left(\frac{a}{c}\right)^2+\omega^2}=-\frac{c}{\omega}+\frac{c^3\omega}{a^2+(c\omega)^2}+\frac{a^2c}{\omega\left[a^2+(c\omega)^2\right]}+$$

$$=\frac{-c\left[a^2+(c\omega)^2\right]+c^3\omega^2+a^2c}{\omega\left[a^2+(c\omega)^2\right]}=\frac{-ca^2-c^3\omega^2+c^3\omega^2+ca^2}{\omega\left[a^2+(c\omega)^2\right]}=0.$$

Множення частини, що залишилася у виразу (11) на експоненту $\exp\left(-\frac{a}{c}x\right)$, згідно (9), приводить до наступного:

$$\frac{c}{\omega}+\frac{c}{\left(\frac{a}{c}\right)^2+\omega^2}\left(\frac{a}{c}\sin\omega x-\omega\cos\omega x\right)-\frac{\frac{a}{\omega}}{\left(\frac{a}{c}\right)^2+\omega^2}\left(\frac{a}{c}\cos\omega x+\omega\sin\omega x\right)=$$

$$=-\frac{c^2}{a^2+(c\omega)^2}\frac{a^2+(c\omega)^2}{c\omega}\cos\omega x.$$

Скорочуючи в даному виразі множники, і після диференціювання згідно (9), відновлюємо функцію в її початковому вигляді: $\psi(x)=\sin\omega x$. Можна помітити, що вона зовсім не схожа на зашифроване повідомлення (10). Таким чином, за рахунок використання розглянутих пропозицій можна досягнути поставленої мети – забезпечення криптографічного шифрування даних, що представлені у формі неперервних однозначних функціональних залежностей, та їх наступне дешифрування, завдяки ефективного вирішення інтегральних рівнянь.

4 Висновки

В межах роботи кількох міжнародних конференцій [5-7] автори доповідали можливі варіанти практичної реалізації даного способу (криптографічних перетворень), крім того був запропонований варіант функціональної системи шифрування, та визначені орієнтовні терміни виготовлення дослідного зразку відповідної криптографічного системи [8]. Як виявилося за результатами проведення профільних наукових дискусій, створення відповідної системи є цілком можливим. Це, принаймні, буде паритетним рішенням по відношенню до потенційних "криптографічних" загроз з боку квантового комп'ютеру. Зрозуміло, що для коректної організації порівняльного експерименту потрібен зразок діючого квантового комп'ютеру, що й обумовлює зупинку досліджень в запропонованому авторами напрямі криптографії.

Посилання

[1] Средства обеспечения информационной безопасности в сетях передачи данных: задачи и методические указания / Составители: А. В. Крыжановский, Н. В. Киреева, В. В. Пугин. – Самара: Поволжская государственная академия телекоммуникаций и информатики, 2008. – 61 с.

[2] Форсайт Дж., Малькольм М. Моулер К. Машинные методы математических вычислений. – М.: Мир, 1980. – 280 с.
[3] Перчик Е. Методология синтеза знаний: преодоление фактора некорректности задач математического моделирования / www.pelbook.narod.ru (2-я ред.)
[4] Полянин А. Д., Манжиров А. В. Справочник по интегральным уравнениям. – М.: Физматлит, 2003. – 608 с.
[5] Громыко И.А. Криптография нового поколения с сопряжением дискрет / И. А. Громыко, К. О. Швагер // Матеріали V-ої Міжнародної НТК «Захист інформації і безпека інформаційних систем». Тез. Доп. – Львів: Вид-во Львівська політехніка. – 2016. – 172 с. – С.104-106. ResearchGate: www.researchgate.net/publication/301747721 - DOI: 10.13140/RG.2.1.3936.8567
[6] Громыко И.А. Постквантовая криптография в ракурсе общей парадигмы защиты информации // Материалы 5-й Международной НТК «Информационные системы и технологии. Харьков – Коблево. ИСТ-2016». Харьков – Коблево. Тезисы доклада. 12 страниц. - Индекс DOI: 10.13140/RG.2.2.29107.02.087.
[7] Громико І.О. Общая парадигма защиты информации в свете новой редакции (2011г.) Закона Украины «Про информацию» // Матеріали Міжнародної НП Інтернет–конференції «Інформаційна і економічна безпека (INFECO-2014).
[8] Громыко И.А. Криптография в общей парадигме защиты информации. Вариант выхода из квантового кризиса // Защита информации. INSIDE. –СпБ.: ООО «Издательский Дом «Афина» -№6 – 2016 г. – с. 48-56.

Reviewer: Volodymyr Maxymovych, Doctor of Sciences (Eng.), Full Prof., ICTA, Lviv Polytechnic National University, Bandera St., 12, Lviv, 79013, Ukraine. E-mail: vmax@polynet.lviv.ua

Received on Jule 2020.

Authors:

Mykhailo Suknov, Cand. Sc. (Education), Prof., Head of department, Kharkiv National University of Radio Electronics, Ukraine.
E-mail: mykhailo.suknov@nure.ua

Igor Gromyko, Ph.D. (Technical), Associate Professor, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: i.gromyko@karazin.ua

Eugene Perchik, Ph.D., Senior Research Officer, Joint Stock Company "STI TRR", Kharkiv, Ukraine.

Method of cryptologic data transformations.

Abstract. Countering a quantum computer in the process of illegal ultra-high-speed decryption of messages is technically feasible. Information owner must oppose the competitor's computer with tasks, the solution of which requires an infinite number of operations during decryption. For example, the dependence of functions on an infinite number of informative features. The owner encrypts by integrating the functions, the recipient decrypts by solving the integral equations. It is not a discrete but an analog approach that prevails here. The basis for the implementation of this approach was created by Polish scientists. Mathematician Stefan Banach (*1892-1945*), who created modern functional analysis, and Marian Mazur (*1909-1983*), the author of " The Qualitative Theory of Information". Their theory was created in contrast with the "Quantitative Information Theory". Cryptologists who have devoted their whole lives to improving the "discrete" theory and found themselves close to power (*and finance*), try not to recall that Claude Shannon in his basic work "Communication Theory of Secrecy Systems" more than once emphasized the discrete focus of his developments anticipating future research on the specific limitations of his work adapted to the communication theory. Forgetting about the unlimited speeds and amounts of memory of quantum computers the orthodox talk about redundancy and further purely technical issues, including administrative leverages for counteracting against opponents. It is impossible to stop the progress of science. Experiments have shown the reality of creating such post-quantum-level cryptographic systems.

Keywords: Cryptology; Quantum Computer; Automated encryption technologies and decryption.

Рецензент: Владимир Максимович, д.т.н., проф., Национальный университет «Львовская политехника», Львов, 79013 Украина. E-mail: vmax@polynet.lviv.ua

Поступила: Июль 2020.

Авторы:

Михайло Сукнов, к.т.н., проф., зав. каф., Харьковский национальный университет радиоэлектроники, 61166, Украина.
E-mail: kate7smith12@gmail.com

Игорь Громыко, к.т.н., проф., Харьковский национальный университет имени В.Н. Каразина, 61022, Украина.
E-mail: i.gromyko@karazin.ua

Евгений Перчик, к.т.н., ст. научный сотрудник, АО НТИ ТТР, Харьков, Украина.

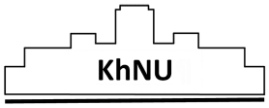
Способ криптологических преобразований данных.

Аннотация. Противодействие квантовому компьютеру в процессе неуполномоченной сверхскоростной расшифровки сообщений - технически осуществимо. Собственник информации должен противопоставить компьютеру конкурента задачи, решение которых потребует бесконечное число операций при дешифровке. Например, - зависимость функций от бесконечного числа информативных признаков. Собственник шифрует путем интегрирования функций, получатель дешифрует ре-

шением интегральных уравнений. В этом случае превалирует не дискретный, а аналоговый подход. Базис для реализации такого подхода создали польские учёные. Математик Стефан Банах (*Stefan Banach, 1892–1945*), создавший современный функциональный анализ и Мариан Мазур (*Marian Mazur, 1909-1983*) автор «Качественной теории информации». Их теория была создана в противовес «Количественной теории информации». Криптологи, которые всю жизнь посвятили совершенствованию «дискретной» теории и оказались приближенными к власти (*и финансам*), стараются не вспоминать, что Клод Шеннон в своей базовой работе «Теория связи в секретных системах» не единожды подчеркнул дискретную направленность своих разработок, предупредив будущих исследователей о специфической ограниченности своей работы, адаптированной под теорию связи. Забывая о неограниченных скоростях и объёмах памяти квантовых компьютеров, ортодоксы говорят про избыточность и прочие сугубо технические моменты, применяя в отношении своих оппонентов все возможные рычаги противодействия. Прогресс науки остановить невозможно. Исследования показали реальность создания подобных криптографических систем постквантового уровня.

Ключевые слова: криптология; квантовый компьютер; технологии автоматизированного шифрования и дешифрования.

ISSN 2519-2310 DOI: 10.26565/2519-2310-2020-2



Наукове видання

КОМП'ЮТЕРНІ НАУКИ ТА КІБЕРБЕЗПЕКА

Випуск 2(18) 2020

Міжнародний електронний науково-теоретичний журнал

Англійською, українською, російською мовами

Комп'ютерне верстання – Федоренко В.В., Єсіна М.В.

61022, Харків, майдан Свободи, 6
Харківський національний університет імені В.Н. Каразіна

V. N. Karazin Kharkiv National University Publishing

CS&CS

2020