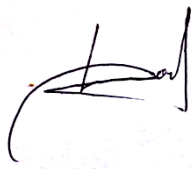
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ДАВИДОВ ВЯЧЕСЛАВ ВАДИМОВИЧ



УДК 004.89+004.942

МОДЕЛІ ТА МЕТОДИ ПІДВИЩЕННЯ БЕЗПЕКИ БАЙТ-КОД ОРІЄНТОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В УМОВАХ КІБЕРАТАК

Спеціальність 05.13.05 – Комп'ютерні системи та компоненти

Автореферат
дисертації на здобуття наукового ступеня
доктора технічних наук

Черкаси – 2021

Дисертацією є рукопис.

Робота виконана на кафедрі обчислювальної техніки та програмування Національного технічного університету «Харківський політехнічних інститут» Міністерства освіти і науки України.

Науковий консультант: доктор технічних наук, професор
Семенов Сергій Геннадійович,
Національний технічний університет «Харківський політехнічний інститут», завідувач кафедри обчислювальної техніки та програмування.

Офіційні опоненти: доктор технічних наук, професор
Коваленко Андрій Анатолійович,
Харківський національний університет радіоелектроніки, Міністерство освіти і науки України, завідувач кафедри електронних обчислювальних машин.

доктор технічних наук, професор
Швачич Генадій Григорович,
Національна металургійна академія України, завідувач кафедри прикладної математики та обчислювальної техніки.

доктор технічних наук, доцент
Бабенко Віра Григорівна,
Черкаський державний технологічний університет, доцент кафедри інформаційної безпеки та комп'ютерної інженерії.

Захист дисертації відбудеться « 09 » вересня 2021 р. о 12.00 на засіданні спеціалізованої вченої ради Д 73.052.04 у Черкаському державному технологічному університеті за адресою: 18006, м. Черкаси, бульвар Шевченка, 460, конференц-зала 1 корпус.

З дисертацією можна ознайомитися в бібліотеці Черкаського державного технологічного університету за адресою: 18006, м. Черкаси, бульвар Шевченка, 460.

Автореферат розіслано « 07 » серпня 2021р.

Учений секретар
спеціалізованої вченої ради Юлія БОНДАРЕНКО

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність. Повсюдне поширення комп'ютерних систем та технологій, збільшення попиту на інформаційні послуги, впровадження обчислювальних та інших засобів обробки даних у ключових сферах життя – все це є основними ознаками сучасного суспільства. В той же час, в останні роки спостерігається збільшення кіберзагроз різної складності. При цьому фахівці відзначають тенденції до росту фінансових, іміджевих, соціальних та інших витрат у зв'язку з їх реалізацією. Невід'ємною складовою існуючих комп'ютерних систем є програмне забезпечення. Проведені дослідження показали, що з точки зору безпеки, програмне забезпечення є однією з найбільш уразливих складових комп'ютерних систем. Пов'язано це з цілою низкою чинників об'єктивного і суб'єктивного характеру (висока вартість втрат і, як наслідок, підвищений інтерес зловмисників, недоліки сучасних засобів і платформ розробки програмного забезпечення, недостатня компетенція призначеного для користування персоналу та ін.).

Аналіз сучасних програмних продуктів показав тенденцію розвитку використання байт-код орієнтованого програмного забезпечення. Це пов'язано з можливістю створення платформо-незалежного коду, а також сучасних та ефективних механізмів роботи з пам'яттю (наприклад, через використання garbage collectors). Завдяки архітектурі платформо-незалежного коду (а саме зберігання проміжного коду, який можна декомпілювати), байт-код орієнтоване програмне забезпечення вразливе до кібератак, що пов'язані з порушенням конфіденційності та автентичності.

Проведені дослідження показали, що багато наукових праць присвячено проблематиці безпеки програмних засобів. Це роботи зарубіжних та вітчизняних науковців, серед яких варто відзначити наступних: Рудницький В., Кузнецов О., Харченко В., Christodorescu M., Denning P., Sheng Z., Шибанов А., Collberg C., Fowler M., Летичевский А., Mohsen R., Schrittwieser S., Roy C., Hua L., Подимов В. Однак, проблеми, пов'язані з безпекою байт-код орієнтованого програмного забезпечення в умовах кібератак на ліцензійні ідентифікатори та обфускацію в цих роботах не розкривались. Але це важливо в умовах підвищених вимог до конфіденційності та автентифікації саме цього виду програмних продуктів.

Таким чином, на сьогоднішній день в теорії і практиці забезпечення безпеки програмних засобів загострилося **протиріччя** між підвищенням вимог до безпеки програмного забезпечення, збільшенням кіберзагроз на послуги автентифікації та конфіденційності та недостатньою якістю сучасних моделей та методів забезпечення безпеки байт-код орієнтованих програмних засобів, які не в змозі забезпечити необхідні якісні характеристики.

Подолати цю суперечність можна шляхом вирішення актуальної **науково-практичної проблеми** підвищення безпеки байт-код орієнтованого програмного коду в умовах кібератак на основі синтезу підсистеми забезпечення конфіденційності та автентичності програмних продуктів.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана у межах пріоритетних наукових напрямів, які охоплюють актуальні проблеми, відповідно до рішення Ради президентів академій наук України від 30

січня 2019 року «Про основні наукові напрями та найважливіші проблеми фундаментальних досліджень у галузі природничих, технічних, суспільних і гуманітарних наук Національної академії наук України на 2019-2023 роки», «Інформатика» за темами: «Розроблення і удосконалення методів верифікації та тестування баз знань», «Розроблення математичних методів та систем моделювання об'єктів та процесів». Дисертаційну роботу виконано у межах зареєстрованих науково-дослідних робіт Національного технічного університету «Харківський політехнічний інститут»: «Дослідження методів управління та захисту даних в комп'ютеризованих інформаційно-вимірювальних та розподілених системах» (ДР №0119U002603) та «Створення завдань по мові програмування "С", тестування методологічної концепції, адаптації програми для інтеграції у систему вищої освіти України» (ДР №0119U103871).

Мета і задачі дослідження. Мета дисертаційної роботи – підвищення безпеки програмного забезпечення в умовах використання байт-код орієнтованих мов програмування. Мета дисертаційної роботи визначає необхідність розв'язання таких **основних задач**:

1. Порівняльний аналіз та дослідження моделей і методів забезпечення безпеки байт-код орієнтованого програмного забезпечення.
2. Розроблення показника безпеки програмних засобів.
3. Розроблення методу перевірки логіко-смислової подібності програм.
4. Розроблення GERT-моделі процесу обфускації програмних модулів з використанням парадигми математичного апарату гамма-розподілу в якості ключового на всіх етапах моделювання процесу обфускації.
5. Розроблення критерію якості обфускації коду для багатопроєктного рішення на основі дослідження та оцінки показників якості коду.
6. Розроблення моделі системи безпеки програмного забезпечення на основі математичного апарату моделювання GERT-мереж з використанням операцій безпечного переходу та кодування ліцензійних ідентифікаторів.
7. Розроблення моделі системи безпеки програмного забезпечення шляхом безпечного переходу і кодування ліцензійних ідентифікаторів математичного апарату моделювання GERT-мереж.
8. Розроблення методу формування цифрового ідентифікатора програмного забезпечення для захисту авторських прав.
9. Розроблення імітаційної моделі систем формування вимог до безпеки та захисту програмного забезпечення.
10. Обґрунтування достовірності одержаних результатів наукових досліджень.
11. Розроблення практичних рекомендацій щодо застосування розробленого методу підвищення безпеки байт-код орієнтованого програмного забезпечення.
12. Дослідження та впровадження розроблених моделей та методів підвищення безпеки байт-код орієнтованого програмного забезпечення в умовах кібер-атак.

Об'єктом дослідження є процес розробки та експлуатації програмного забезпечення.

Предметом дослідження є моделі і методи підвищення безпеки байт-код орієнтованого програмного забезпечення.

Методи дослідження. Для вирішення завдань математичного моделювання процесів обфускації використано теорію графічної оцінки та аналізу, теорію графів та теорію ймовірностей. Для створення програмної імітаційної моделі використано методи об'єктно-орієнтованого програмування та методи роботи з нереляційними базами даних. Для синтезу підсистем забезпечення конфіденційності та автентичності програмних продуктів було використано теорію статистичної обробки даних, теорію інформаційної безпеки та теорію технічного аналізу.

Наукова новизна одержаних результатів полягає у наступному:

– *Отримав подальший розвиток* метод перевірки логіко-смислової подібності програм складної логічної структури, що відрізняється від відомих розпаралелюванням процесів, що обчислюються при порівнянні подібних елементів програмного коду, а також формуванням та використанням графу спільних обчислень в процесі пошуку подібних елементів коду. Це дозволило знизити складність процесу верифікації.

– *Вперше розроблена* GERT-модель процесу обфускації програмних модулів, що реалізує парадигми використання математичного апарату гамма-розподілу у якості ключового на всіх етапах моделювання процесу обфускації. Це дозволило досягти уніфікації моделі в умовах модифікації GERT-мережі.

– *Вдосконалено метод* обфускації програмних модулів, що відрізняється від відомих урахуванням варіативності типів лексем та ідентифікаторів. Це дозволило підвищити показник безпеки програмного забезпечення.

– *Вперше розроблено критерій* оцінки якості обфускації програмного забезпечення шляхом синтезу лінійної композиції часткових критеріїв метрик якості коду, що дозволило кількісно оцінити ступінь обфускованості програмних продуктів.

– *Вперше розроблено* модель безпечного переходу і кодування ліцензійних ідентифікаторів на основі математичного апарату GERT-мереж з парадигмою гамма-розподілу, що дозволило підвищити точність результатів моделювання.

– *Вдосконалено* метод формування цифрового ідентифікатора програмного забезпечення для захисту його авторських прав шляхом введення та вдосконалення модулів менеджерів ліцензій, контроля цілісності та ідентифікації. Відмінною особливістю даної моделі є використання індивідуальних даних комп'ютерної системи кінцевого користувача для однозначної ідентифікації приналежності, що дозволило підвищити безпеку байт-код орієнтованих програмних застосунків в умовах кібератак.

Практичне значення одержаних результатів. Отримані в дисертаційній роботі результати дають змогу підвищити безпеку байт-код орієнтованих програмних додатків, що в свою чергу дозволяє забезпечити достатній рівень захищеності програмного забезпечення в умовах кібер-атак.

Практична цінність роботи полягає у наступному:

– розроблено метод перевірки логіко-смислової подібності програм складної логічної структури для зменшення часу процесу верифікації;

– розроблено алгоритми обфускації програмного коду для підвищення захисту коду від реверс-інжинірингу;

– розроблено критерій оцінки якості обфускації програмного коду для підвищення точності оцінки безпеки програмного забезпечення;

– розроблено алгоритми формування персоналізованого ліцензійного ключа для захисту програмного забезпечення від неліцензійного копіювання;

– розроблено алгоритми захисту ліцензійних ключів від копіювання на основі прихованих переходів та кодування ліцензійного ключа.

Практичне значення отриманих результатів підтверджено відповідними актами впровадження.

Результати дисертації впроваджені і використовуються у діяльності компаній «Line Up», «Нікс Солюшенс ЛТД», Державного підприємства «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості», Державного підприємства «Харківський науково-дослідний інститут технологій машинобудування», а також використано у навчальному процесі Національного технічного університету «Харківський політехнічний інститут».

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи автор отримав самостійно. Вони викладені як в роботах, які опубліковані без співавторів [20, 24, 30], так і у співавторстві. У друкованих працях, опублікованих у співавторстві, здобувачеві належать: [15, 28] – дослідження вимог безпеки до програмних засобів; [3, 7, 14, 19, 21, 26, 35] – дослідження методів виявлення інформаційних та кібератак та впливів на системи обробки даних; [22] – розробка методу оцінки ступеню обфускованості коду; [23, 33] – розробка методу захисту програмного коду у вбудованих системах; [1, 16, 17] – дослідження методу графічної оцінки та аналізу для створення методів підвищення безпеки програмного забезпечення; [11, 25] – дослідження методів захисту програмного забезпечення; [9] – розробка методу обфускації програмного забезпечення; [2] – розробка способу побудови синтаксичного дерева коду; [4, 5, 27, 29] – розробка методу генерації ліцензійного ключа на основі даних комп'ютерної системи кінцевого користувача; [6, 8, 10, 12, 13, 18, 31, 32, 34, 36] – дослідження методів функціонування, масштабування та безпеки систем хмарних обчислень.

З робіт, що опубліковані у співавторстві, у дисертаційній роботі використовуються виключно результати, отримані особисто здобувачем.

Апробація результатів дисертації. Основні положення дисертаційної роботи доповідалися та обговорювалися на таких наукових конференціях та семінарах: XXIII Міжнародна науково-практична конференція «Інформаційні технології, наука, техніка, технологія, здоров'я» (Харків, 2015); XV Міжнародний науковий семінар «Сучасні проблеми інформатики в управлінні, економіці, освіті та подоланні наслідків Чорнобильської катастрофи» (Шацьк, 2016); 26th National Scientific Symposium with International Participation «Metrology and Metrology assurance» (Sozopol, Bulgaria, 2016); 7th World Congress «Aviation in the XXI-st Century» (Київ, 2016); VII Міжнародна науково-технічна конференція «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (Полтава, 2017); V Міжнародна науково-технічна конференція «Проблеми інформатизації» (Полтава, 2017); 28th International Scientific Symposium «Metrology and Metrology Assurance (MMA)» (Sozopol, Bulgaria, 2018); VI Міжнародна науково-технічна конференція

«Проблеми інформатизації» (Черкаси, 2018); Всеукраїнська науково-практична конференція «Актуальні питання протидії кіберзлочинності та торгівлі людьми» (Харків, 2018); 10th International Conference on Dependable Systems, Services and Technologies (DESSERT) (Leeds, UK, 2019); 29th International Scientific Symposium «Metrology and Metrology Assurance (MMA)» (Sozopol, Bulgaria, 2019); Sun SITE Central Europe (CEUR) Workshop Proceedings (Kyiv, 2019); VIII Міжнародна науково-технічна конференція «Проблеми інформатизації» (Черкаси, 2020); XX Міжнародна науково-практична конференція «Інформаційні технології і безпека» (Київ, 2020); XI Міжнародна науково-технічна конференція «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (Харків, 2021), Fifth International Scientific and Technical Conference «Computer and Information Systems and Technologies» (Kharkiv, 2021).

Публікації. Основні положення дисертації опубліковано в 36 наукових працях, серед яких: 18 наукових статей (з них: 1 входить до бази даних Scopus (другий квартіль); 1 - у закордонному рецензованому виданні; 16 - у вітчизняних фахових наукових журналах), 16 тез доповідей (з них: 3 входять до бази даних Scopus), а також 2 монографії (з них – 1 одноосібна).

Структура роботи та її обсяг. Дисертація складається зі вступу, шести розділів, загальних висновків, списку використаної літератури та додатків і містить 244 сторінок основного тексту, 49 рисунків, 14 таблиць, 243 джерел у списку літератури, 69 сторінок додатків. Загальний обсяг роботи 313 сторінок.

ОСНОВНА ЧАСТИНА

У **вступі** подано загальну характеристику роботи, обґрунтовано актуальність, сформульовано мету і задачі досліджень, відображено наукову новизну і практичну цінність отриманих результатів, наведено дані щодо їх апробації та впровадження.

У першому розділі проведено аналіз наукової літератури за темою дисертаційної роботи.

У розділі описаний життєвий цикл програмного продукту. Показано доцільність приділення уваги стадії аналізу вимог до програмних засобів, а також їх адаптації з урахуванням виявленої специфіки програмного забезпечення, методологій розробки, засобів розробки та інших факторів.

Класифіковано вимоги до програмного забезпечення. Показано, що практична реалізація поставлених завдань підвищення захищеності (як однієї з характеристик якості ПЗ) може спричинити за собою погіршення інших показників якості ПЗ, наприклад, таких як переносимість, продуктивність, супровідність.

Наведено метрики захищеності, що охоплюють досліджувані характеристики, а також метрики характеристик якості ПЗ, які мають негативний вплив на метрики захищеності.

На основі проведених досліджень сформовано матрицю впливу моделей і методів, що підвищують захищеність ПЗ на метрики інших характеристик якості ПЗ. Дані матриці представлені в табл. 1.

Таблиця 1. Вплив базових характеристик якості програмного забезпечення на характеристики його захищеності

	Здатність до відстеження доступів	Контрольованість доступу	Запобігання спотворенню даних
Здатність до аналізованості	+	—	—
Складність модифікації	+	+/-	+
Ступінь узгодженості і супроводжуваності	+	+/-	—
Продуктивність	+	+	+

Таким чином, актуалізується необхідність вирішення завдання підвищення безпеки ПЗ з одночасним дотриманням вимог до основних показників його якості.

Проведені дослідження показали, що всі джерела загроз безпеці інформації можна розділити на три основні групи: обумовлені діями суб'єкта (антропогенні джерела загроз); обумовлені технічними засобами (техногенні джерела небезпеки); обумовлені стихійними джерелами. При цьому щодо програмного забезпечення найбільш істотними є антропогенні джерела загроз.

Дані джерела загроз мають різну ступінь небезпеки, яку можна кількісно оцінити, провівши їх ранжування. Ці джерела впливають на показник безпеки програмного забезпечення $K_{без}$.

Розроблена функція безпеки програмного забезпечення, перевагою якої є мінімізація впливу коефіцієнтів, що несуть суб'єктивний характер:

$$K_{без} = \sqrt[n]{\frac{1\Big/\prod_{1}^{n} k_i + \sum_{1}^{n} k_i}{1\Big/\sum_{1}^{n} k_i}},\ K_{без} \to \max, k_i \in [0..1], \qquad (1)$$

де n – кількість характеристик, що описують безпеку програмного забезпечення. У нашому випадку використовується 3 характеристики;

k_1 – можливість виникнення джерела загрози. Дана характеристика безпосередньо залежить від рівня захищеності програмного забезпечення:

$$k_1 = f(k_c, k_i, k_a), k_1 \to 1; \qquad (2)$$

k_2 – інвестиційна привабливість. Дана характеристика описується наступними показниками: фінансова привабливість програмного продукту, інноваційна привабливість, соціальна привабливість, інформаційна привабливість:

$$k_2 = A = f(A_f, A_{it}, A_s, A_{in}); \qquad (3)$$

k_3 – ступінь кваліфікації суб'єкта атаки. Суб'єкт атаки, зловмисник, характеризується: рівнем підготовки, ступенем обізнаності про внутрішню структуру ПЗ, переслідуваними цілями:

$$k_3 = Q = f(Q_p, Q_i, Q_m); \qquad (4)$$

Експертним шляхом та на основі аналізу переваг та недоліків сучасних програмних продуктів, були обчислені значення коефіцієнтів для досліджуваних характеристик. Результати дослідження представлені в табл. 2 для двох основних категорій програмних продуктів (Standalone-застосунки, Web-застосунки).

Таблиця 2 – Показник безпеки для Standalone і Web застосунків

Тип ПЗ	k_1	k_2	k_3	$К_{без}$
Standalone-застосунки	0.4	1	0.59	2.01
Web-застосунки	0.71	0.66	1	2.20

За представленими у табл. 2 даними можна зробити висновок про доцільність пріоритетності забезпечення безпеки саме Standalone-застосунків, так як значення показника безпеки там нижче.

Проаналізовані програмні продукти дозволили виділити 5 рівнів проектування: прикладний, системний, ресурсний, операційний.

Аналіз системного рівня проектування показав, що прямий доступ до виконуваного коду програмного продукту для Standalone-застосунків дозволяє використовувати механізми трасування і байт-код маніпуляцій, які дозволяють порушити такі послуги безпеки програмного продукту, як цілісність, конфіденційність і управління доступом.

Проведені дослідження показали, що для Standalone-застосунків саме системний рівень є одним з найбільш вразливих. Саме на цьому рівні можливі злочинні дії, спрямовані на дискредитацію таких послуг безпеки, як: конфіденційність, цілісність, непідробленість, справжність, захищеність від помилки.

Як результат, в дисертаційній роботі сформульовано завдання дисертаційного дослідження, в якому визначено, що основним завданням розроблення моделей та методів підвищення безпеки байт-код орієнтованого програмного забезпечення в умовах кібератак є удосконалення і вибір моделей, методів і засобів, що підвищують рівень конфіденційності, цілісності, непідробленості, справжності, захищеності від помилки.

У другому розділі проаналізовано алгоритм перевірки логіко-смислової подібності стандартних послідовних схем програм, заснований на пошуку найбільш схожих з точки зору лексемної історії маршрутів в програмі. Цей алгоритм використовує в якості опорної структури граф спільних обчислень.

Нехай задані дві програми $p' = \langle X, Y', V', v_{in}', v_{out}', B', \to', L_0' \rangle$ і $p'' = \langle X, Y'', V'', v_{in}'', v_{out}'', B'', \to'', L_0'' \rangle$. Вважаємо, що зазначені дві програми мають однакову множину вхідних змінних X і непересічні множини внутрішніх змінних $Y' \cap Y'' = \emptyset$.

$\Gamma_{p_1,p_2} = \langle v, w_0, \mapsto, L_0 \rangle$ має наступні наступними компоненти: $V = V' \times V''$ – множина вузлів графа; $w_0 = (v_{in}', v_{in}'')$ – виділена коренева вершина; $\to \subseteq V \times \{0,1\} \times Заміна(Y' \cup Y'', Y' \cup Y'', F) \times V$ – відношення переходів, що визначає дуги графа; $L_0 = L_{0'} \cup L_{0''}$ – ініціалізуюча заміна графа спільних обчислень програм.

Замінимо $(v', v'', d, Q, u', u'') \in \mapsto$ наступним чином: $(v', v'') \overset{d,Q}{\mapsto} (u', u'')$.

Відношення $\mapsto$: для кожної пари вузлів (v', v''), (u', u'') графу $\Gamma_{p',p''}$ і заміни $Q, Q \in Заміна(Y' \cup Y'', Y' \cup Y'', F)$, має бути виконано:

$$(v', v'') \overset{d,Q}{\mapsto} (u', u'') \Leftrightarrow \exists d \in \{0,1\} : v' \overset{d,Q'}{\to} u' \& v'' \overset{d,Q''}{\to} u'' \& Q = Q' \cup Q'' \qquad (5)$$

Введено поняття шляху в графі спільних обчислень: будь-яка послідовність дуг:

$$шлях = (v'_{in}, v''_{in}) \overset{d_0,Q_0}{\mapsto} (v'_1, v''_1) \overset{d_1,Q_1}{\mapsto} \dots \overset{d_{n-1},Q_{n-1}}{\mapsto} (v'_n, v''_n) \overset{d_n,Q_n}{\mapsto} (v'_{n+1}, v''_{n+1}) \quad (6)$$

є шляхом у графі спільних обчислень Γ_{p_1,p_2}. Для кожного шляху у графі Γ_{p_1,p_2} введемо поняття заміни маршруту $Q_{шлях}$, що являє собою композицію ініціалізуючої заміни L_0 і всіх замін, що приписані дугам цього шляху: $Q_{шлях} = L_0 Q_1 Q_2 \dots Q_{n-1} Q_n$.

Два маршрута tr' і tr'' в програмах p' і p'' відповідно є узгодженими, якщо вірно, що

$$tr' = v'_{in} \overset{d_0,Q'_0}{\to} v'_1 \overset{d_1,Q'_1}{\to} \dots \overset{d_n,Q'_n}{\to} v'_{n+1},\ tr'' = v''_{in} \overset{d_0,Q''_0}{\to} v''_1 \overset{d_1,Q''_1}{\to} \dots \overset{d_n,Q''_n}{\to} v''_{n+1} \quad (7)$$

Вектор $\tilde{d} = (d_0, d_1, \dots, d_n)$ є загальним вектором узгоджених маршрутів tr', tr''. Шлях в графі спільних обчислень, що проходить по вершинах

$$шлях = (v'_{in}, v''_{in}) \overset{d_0,Q'_0 \cup Q''_0}{\mapsto} (v'_1, v''_1) \overset{d_1,Q'_1 \cup Q''_1}{\mapsto} \dots \overset{d_n,Q'_n \cup Q''_n}{\mapsto} (v'_{n+1}, v''_{n+1}) \quad (8)$$

відповідний парі узгоджених маршрутів tr' і tr'' за вектором d.

Для всіх вузлів $w, w \in V$ графа спільних обчислень Γ_{p_1,p_2} використовуватиме позначення $Шлях(w)$ для множини всіх шляхів графа, що ведуть із вхідного вузла w_0 у вузол w.

Із введених вище визначень графа спільних обчислень, узгоджених маршрутів, а також з умови (5) випливає справедливість наступних лем.

Лема 1. Нехай tr' і tr'' – узгоджені маршрути в програмах p' і p'' відповідно. Тоді в графі спільних обчислень Γ_{p_1,p_2} існує єдиний шлях, відповідний парі маршрутів tr', tr''.

Лема 2. Нехай *шлях* – шлях в графі спільних обчислень Γ_{p_1,p_2} програм p' і p''. Тоді у вхідних програмах існує єдина пара маршрутів tr', tr'', якій відповідає цей шлях.

Граф спільних обчислень коректний, якщо в ньому: 1) із вхідних вузлів недосяжні вузли виду (v'_{out}, v''_i) і вузли виду (v'_j, v''_{out}), де $v''_i \neq v''_{out}$ і $v'_j \neq v'_{out}$ відповідно; 2) із кожного вузла графу досяжний вузол (v'_{out}, v''_{out}).

З лем 1 та 2 доведена наступна теорема. Теорема 1. Дві програми $p' = \langle X, Y', V', v_{in}', v_{out}', B', \to', L_0' \rangle$ і $p'' = \langle X, Y'', V'', v_{in}'', v_{out}'', B'', \to'', L_0'' \rangle$ подібні за логікою та сенсом тоді і лише тоді, коли їх граф спільних обчислень коректний і для будь-якого шляху в ньому: $шлях = (v'_{in}, v''_{in}) \overset{d_0,Q'_0 \cup Q''_0}{\mapsto} (v'_1, v''_1) \overset{d_1,Q'_1 \cup Q''_1}{\mapsto} \dots \overset{d_n,Q'_n \cup Q''_n}{\mapsto} (v'_{n+1}, v''_{n+1})$ виконано рівність $B'(v'_{n+1}) Q_{tr'} = B''(v''_{n+1}) Q_{tr''}$, де $Q_{tr'}$ і $Q_{tr''}$ – заміни маршрутів tr' і tr'' відповідно таких, що шлях з ними погоджено.

Таким чином, перевірка логіко-смислової подібності може бути зведена до аналізу пар узгоджених маршрутів вихідних програм.

Лема 3. Нехай Q_1 і Q_2, $Q_1, Q_2 \in Заміна(X, Y, F)$ – деякі заміни, а атоми A' і A'', $A', A'' \in Atom(Y, F)$ – умовні вирази. Тоді: $\begin{cases} A'Q_1 = A''Q_1 \\ A'Q_2 = A''Q_2 \end{cases} \Leftrightarrow A'(Q_1 \downarrow Q_2) = A''(Q_1 \downarrow Q_2)$.

Доведення леми 3 дозволяє переформулювати необхідну і достатню умову логіко-смислової подібності програм, наведену в теоремі 1, в наступний спосіб.

Теорема 2. Дві програми p' і p'' логіко-смислової подібні тоді і лише тоді, коли граф їх спільних обчислень коректний і для кожної його вузла $w = (v', v'')$ виконано рівність $B'(v')Q_w = B''(v'')Q_w$, де $Q_w = \downarrow_{шлях \in Шлях(w)} Q_{шлях}$.

Наведена теорема зводить задачу про перевірку подібності програм до обчислення точних нижніх меж замін, відповідних всіляким шляхам в графі спільних обчислень, ведучим в кожний з вузлів w графу.

В розділі наведена процедура глобальної розмітки графа, що дозволяє вирішувати цю задачу. Ця процедура складається з наступних дій:

– перейменування входжень всіх змінних з множини X в першу програму змінними $\{x'_1, x'_2, \ldots, x'_n\}$, відмінними від змінних множини X. Перейменування входжень всіх змінних з множини X в другу програму змінними $\{x''_1, x''_2, \ldots, x''_n\}$, також відмінними від змінних множини X;

– для програм p_1, p_2 будується їх граф спільних обчислень Γ_{p_1, p_2};

– виконується початкова розмітка графа;

– виконується ітерація по графу Γ_{p_1, p_2}, доки існують активні вершини;

– верифікація. Здійснюється перевірка наступної умови: чи існує в графі Γ_{p_1, p_2} хоча б один вузол (u', u''), позначений заміною $n_{(u',u'')}$ такою, що $A_{u'} n_{(u',u'')} \neq A_{u''} n_{(u',u'')}$. Наявність таких вузлів свідчить про відсутність логіко-смислової подібності між програмами.

Нехай $n_v^{[n]}$ – заміна, приписана вузлу $v = (v', v'')$ на n -му кроці алгоритму. Тоді справедливе наступне твердження.

Лема 4. Для будь-якого вузла $v = (v', v'')$ і для будь-якого кроку роботи алгоритму n існує така підмножина P множини $Шлях(v)$, що для заміни $n_v^{[n]}$, обчисленої на етапі n для вузла v, виконано рівність: $n_v^{[n]} = \downarrow_{шлях \in P} Q_{шлях}$.

Для кожного вузла v позначимо $In(v)$ множину всіх вузлів u таких, що $v = succ(u, d)$. При цьому, перехід $u \overset{d,Q}{\mapsto} v$, позначимо Q_{uv}.

Тоді з леми 4 слідує: Лема 5. Для будь-якого вузла v:

$$\downarrow_{шлях \in Шлях(v)} Q_{шлях} = \downarrow_{u \in In(v)} Q_{uv} \left(\downarrow_{шлях \in Шлях(u)} Q_{шлях} \right). \qquad (9)$$

Лема 6. Припустимо, що на певному кроці з номером N алгоритм завершив свою роботу. Тоді $\left\{ n_v^{[N]} : v \in V \right\}$ – рішення системи рівнянь леми 5.

Лема 7. Припустимо, що алгоритм зупинився на кроці N. Тоді для будь-якої вершини v графу спільних обчислень вірна рівність: $n_v^{[N]} = \downarrow_{шлях \in Шлях(v)} Q_{шлях}$.

Наведені леми дозволяють обґрунтувати коректність алгоритму перевірки логіко-смислової подібності програм.

Теорема 3. Для будь-яких двох програм p_1 і p_2 таких, що їх граф спільних обчислень коректний, алгоритм побудови розмітки графу спільних обчислень завжди завершує свою роботу.

Теорема 4. Для двох програм p_1 і p_2, чий граф спільних обчислень коректний, зупинка алгоритму перевірки логіко-смислової подібності програм з позитивним результатом відбувається лише тоді, коли програми p_1 і p_2 – логіко-смислово подібні.

В основі наведеного алгоритму лежить ітеративна процедура, яка на кожному кроці обчислює заміну: кожен раз для деякої трійки підстановок Q_1, Q_2, Q_3 обчислюється заміна $(Q_1Q_2)\downarrow Q_3$. Відповідно до леми 3, розмір графа, що реалізує цю заміну, можна оцінити величиною, пропорційною розмірам графів вихідних замін. У гіршому випадку, наведений алгоритм повинен здійснити кількість ітерацій, пропорційну розмірам вихідних програм. Доведено, що цей алгоритм має експонентну складність. Для того, щоб усунути цей недолік алгоритму, розглянута операція – деуніфікація замін. Ця операція обчислює нижню межу двох замін. Вона здійснюється за лінійний час, а орієнтований граф, що представляє її результат, за розміром не перевищує мінімальний з графів вихідних замін.

Заміна Q, $Q \in Заміна(X,Y,F)$ є скороченою, якщо для неї виконано рівність: $Змінна_Q = НЗмінна_Q$. З цього слідують наступні характеристичні властивості скорочених замін:

– в орієнтованому графі, що реалізує скорочену підстановку, всі термінальні вузли озаглавлені;

– кожна заміна Q має хоча б один скорочений прототип. Так, наприклад, порожня заміна $\left\{\frac{x_1}{y_1}, \frac{x_2}{y_2}, \ldots, \frac{x_n}{y_n}\right\}$ є скороченою.

Нехай задана деяка заміна $\theta = \left\{\frac{x_1}{t_1}(\ldots), \frac{x_2}{t_2}(\ldots), \ldots, \frac{x_n}{t_n}(\ldots)\right\}, Q \in Заміна(X,Y,F)$. Процесом скорочення заміни Q наступна послідовність дій. Виберемо з множини Y змінну y таку, що $y \notin Змінна_Q$. Припустимо, що входить в l -у зв'язку заміни Q лексема t_l містить підлексему $f(y_1, y_2, \ldots, y_n)$ таку, що хоча б для однієї j, $1 \le j \le k$, вірно, що $y_i \notin НЗмінна_Q$. Для кожної t_i, $1 \le i \le n$, побудуємо лексему t'_i відповідно до одного з наступних правил:

– якщо t_i не містить підлексему $f(y_1, y_2, \ldots, y_n)$, то $t'_i = t_i$;

– якщо t_i містить підлексему $f(y_1, y_2, \ldots, y_n)$, то t'_i виходить з t_i синхронною заміною всіх входжень підлексем $f(y_1, y_2, \ldots, y_n)$ на змінну y.

Розглянемо заміну $\theta' = \left\{\frac{x_1}{t'_1}(\ldots), \frac{x_2}{t'_2}(\ldots), \ldots, \frac{x_n}{t'_n}(\ldots)\right\}$. З побудови t'_i видно, що $Q = Q'\{y / f(y_1, \ldots y_k)\}$. Крім того, $Змінна_{Q'} = Змінна_Q \cup \{y\}$.

Для побудованої таким чином пари замін Q і Q', і довільної скороченої заміни n справедливо наступне твердження.

Лема 8. Скорочена заміна n є прототипом заміни Q в тому і лише тому випадку, коли n є прототипом заміни Q'.

Найбільш спеціальним скороченням заміни Q є така скорочена заміна Q', що Q – приклад заміни Q' і будь-яка скорочена заміна, що є прототипом Q, буде

прототипом заміни Q'. Для найбільш спеціального скорочення введено позначення: $Q' = нсс(Q)$.

Лема 9. Для кожної заміни Q, $Q \in Заміна(X,Y,F)$ існує її найбільш спеціальне скорочення $нсс(Q)$. Найбільш спеціальне скорочення єдине з точністю до перейменування змінних.

Наведені важливі властивості найбільш спеціальних скорочень замін. Обґрунтування цих властивостей спирається на запропонований алгоритм побудови найбільш спеціальних скорочень замін.

Лема 10. Для будь-яких двох замін Q_1 і Q_2 таких, що $Q_1 \le Q_2$, вірно, що їх скорочення знаходяться в співвідношенні $нсс(Q_1) \le нсс(Q_2)$.

Лема 11. Нехай $Q \in Заміна(X,X,F)$, $n \in Заміна(X,Y,F)$. Тоді справедливе наступне співвідношення: $нсс(Qn) \sim нсс(Qнсс(n))$.

Теорема 4. Для будь-якої пари атомарних формул A, B, $A, B \in Atom(X,F)$ і для будь-якої заміни Q, $Q \in Заміна(X,Y,F)$ співвідношення $AQ = BQ$ справедливе тоді і лише тоді, коли $Ансс(Q) = Внсс(Q)$.

Введена операція скороченої деуніфікації замін, яка дозволяє застосовувати апарат скорочених замін до вирішення завдання перевірки логіко-смислової подібності програм: скороченою деуніфікацією замін Q_1, Q_2, $Q_1, Q_2 \in Заміна(X,Y,F)$ є така заміна Q, що $Q = нсс(Q_1 \downarrow Q_2)$. Для її позначення введемо запис: $Q = Q_1 \Downarrow Q_2$.

З урахуванням леми 10, доведено наступну теорему.

Теорема 5. Для будь-якої пари замін n_1, n_2 таких, що $n_1, n_2 \in Заміна(Y,Y,F)$ справедлива рівність: $n_1 \Downarrow n_2 \sim нсс(n_1) \Downarrow нсс(n_2)$.

Наступні теореми є аналогами теореми 1 і леми 3 для скороченої деуніфікації замін.

Теорема 6. Для будь-якої заміни Q, $Q \in Заміна(X,X,F)$, і пари замін n_1, n_2, $n_1, n_2 \in Заміна(X,Y,F)$ справедливо $Qn_1 \Downarrow Qn_2 = нсс\left(Q\left(n_1 \Downarrow n_2\right)\right)$.

Теорема 7. Для будь-якої пари атомів A', A'' таких, що $A', A'' \in Atom(X,F)$ і будь-якої пари замін n_1, n_2, $n_1, n_2 \in Заміна(X,Y,F)$ справедливе наступне співвідношення:

$$\begin{cases} A'n_1 = A''n_1 \\ A'n_2 = A''n_2 \end{cases} \Leftrightarrow A'\left(n_1 \Downarrow n_2\right) = A''\left(n_1 \Downarrow n_2\right) \qquad (10)$$

Визначення скороченої деуніфікації лексем формально задає алгоритм побудови скороченої нижньої межі замін як послідовного застосування операції деуніфікації і алгоритму побудови найбільш спеціального скорочення. Але тоді обчислення скороченої нижньої межі і розмір результуючої заміни матимуть квадратичну залежність від розмірів орієнтованих графів вхідних замін. Така залежність призведе до того, що алгоритм перевірки логіко-смислової подібності виявиться експоненційним. Тому побудовано алгоритм обчислення скороченої деуніфікації, що має лінійну складність.

Нехай задані дві заміни θ_1, θ_2 такі, що $Q_1, Q_2 \in \textit{Заміна}(X, Y, F)$. Нехай орієнтовані графи Γ_{Q_1} і Γ_{Q_2} реалізують відповідно заміни Q_1 і Q_2. Для заміни $Q = Q_1 \Downarrow Q_2$ буде побудований орієнтований граф Γ_Q, вузлами якого будуть впорядковані пари $w = (u, v)$, де u є вузол графу Γ_{Q_1}, а v – відповідно вузол графа Γ_{Q_2}.

Лема 12. Кількість кроків алгоритму побудови скороченої деуніфікації не більше, ніж $O\left(|X| + \min\left(|\Gamma_{Q_1}|, |\Gamma_{Q_2}|\right)\right)$.

Лема 13. Нехай $Q = Q_1 \Downarrow Q_2$, і при цьому $l(Q) = l(Q_i)$, $i \in \{1, 2\}$. Тоді $l(Q) \le \min\left(l(Q_1), l(Q_2)\right)$, а $Q \sim Q_i$.

Наведені вище твердження дозволяють застосувати алгоритми, що працюють із скороченими замінами, до перевірки логіко-смислової подібності програм.

Леми 4-7, а також теореми 3, 4 з усіх властивостей операції взяття точної нижньої межі замін спираються лише на властивості, показані в лемах 1 і 3. Але справедливість цих же властивостей для операції скороченої деуніфікації показана в лемі 11 і теоремах 4-7. Це означає, що зазначені леми зберігають справедливість всіх тверджень, що обґрунтовують коректність алгоритму перевірки логіко-смислової подібності.

Наведені в лемах 12 і 13 оцінки часу виконання скороченої деуніфікації і розміру її результату дозволяють привести оцінку трудомісткості модифікованого алгоритму перевірки логіко-смислової подібності програм.

Наведений алгоритм працює з графом спільних обчислень вхідних програм, для якого кількість його вузлів свідомо не перевищує величини n^2. Розмір кожного приписаного йому вузла заміни $n_{(v',v'')}$ може бути оцінений зверху величиною n^2. З леми 13 слідує, що на кожному кроці розмір хоча б однієї з замін, приписаних до вузлів графу спільних обчислень, зменшується. Це означає, що кількість ітерацій алгоритму може бути оцінена величиною $O\left(n^4\right)$.

На кожній ітерації алгоритм здійснює процедуру обчислення скороченої деуніфікації замін $Qn_{(v',v'')} \Downarrow n_{(u',u'')}$. Лема 12 показує, що складність здійснення цієї процедури лінійно залежить від розмірів замін. Тоді складність всього алгоритму можна оцінити $O\left(n^6\right)$.

Таким чином, доведено можливість знизити складність процесу верифікації обфускації програмних модулів.

У **третьому розділі** представлено результати дослідження і алгоритми обфускації програмного коду, а саме: строкових виразів, імен ідентифікаторів, логічних виразів.

Розроблено комплекс математичних моделей процесу обфускації. В основу математичного моделювання покладено підхід мережевого GERT-моделювання. В результаті розроблено математичні моделі процесу обфускації програмного коду.

Відповідно до представленого опису розроблено мережеву GERT-модель процесу обфускації програмного коду, графічне зображення якої представлено на рис. 1.

У представленій мережі вузли графа інтерпретуються станами програмного продукту в процесі обфускації, а гілки графа – ймовірнісно-часовими характеристиками переходів між станами. Зокрема, гілка (1,2) описує процес обробки вихідного коду шляхом кодування строкових літералів, вбудовування у вихідний код функції, яка «на льоту» декодує змінені рядкові літерали в початковий стан. Гілка (2,3) відображає процес обфускації булевих операцій. Гілка (3,4) відображає процес обфускації імен ідентифікаторів. Гілка (4,5) характеризує підготовку обфускованого коду до компіляції. Гілка (1,3) характеризує пропуск обробки вихідного коду шляхом кодування строкових літералів і виконання процесу обфускації булевих операцій. Гілка (1,4) характеризує пропуск обробки вихідного коду шляхом кодування строкових літералів і обфускації булевих операцій; і виконання процесу обфускації імен ідентифікаторів. Гілка (2,4) характеризує пропуск обробки вихідного коду шляхом кодування обфускації булевих операцій та виконання процесу обфускації імен ідентифікаторів. Гілка (3,5) характеризує пропуск обробки вихідного коду шляхом обфускації імен ідентифікаторів та підготовка коду до компіляції. Гілка (2,5) характеризує пропуск обробки вихідного коду шляхом обфускації булевих операцій і імен ідентифікаторів та підготовка коду до компіляції. Гілка (2,1) описує виникнення помилки в процесі кодування шляхом видозміни строкових літералів і додавання функції декодування; при цьому виконуємо повернення на початковий стан з метою повторити спробу обфускації. Гілка (3,1) описує виникнення помилки в процесі верифікації успішності перетворення булевих операцій; при цьому виконуємо повернення на початковий стан (так як невідома причина виникнення помилки – проблема обфускації булевих операцій або помилка викликана видозміною строкових літералів на попередніх кроках) з метою повторити спробу обфускації. Гілка (4,1) описує виникнення помилки в процесі верифікації успішності перетворення імен ідентифікаторів; при цьому виконуємо повернення на початковий стан з метою повторити спробу обфускації.

Характеристики гілок моделі представлені в табл. 3. В даній GERT-мережі процесів обфускації програмних модулів функції щільності ймовірності переходів визначимо гамма-розподілом з змінними коефіцієнтами k та θ:

$$\zeta(x) = \frac{x^{k-1} \cdot e^{\frac{-x}{\theta}}}{\theta^k \cdot \Gamma(k)}. \qquad (11)$$

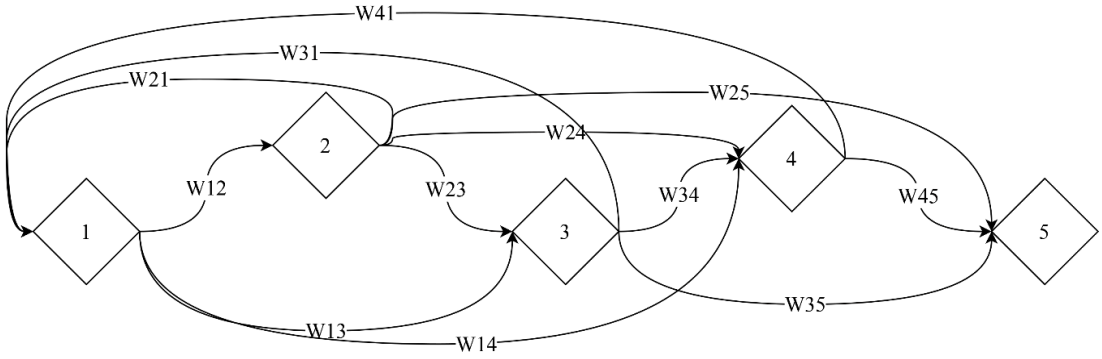


Рисунок 1 – GERT-мережа процесів обфускації програмних модулів

Еквівалентна *W*-функція часу виконання алгоритмів обфускації програмного коду дорівнює:

$$W_E(s)=\frac{W_{12}W_{25}+W_{13}W_{35}+W_{14}W_{45}+W_{12}W_{23}W_{35}+W_{12}W_{24}W_{45}+W_{13}W_{34}W_{45}+W_{12}W_{23}W_{34}W_{45}}{1-\left(W_{12}W_{21}+W_{13}W_{31}+W_{14}W_{41}+W_{12}W_{23}W_{31}+W_{12}W_{24}W_{41}+W_{13}W_{34}W_{41}+W_{12}W_{23}W_{34}W_{41}\right)}, \quad (12)$$

Таблиця 3 – Характеристики гілок (переходів між станами) GERT-мережі процесів обфускації програмних модулів

№ з/п	Гілка	*W*-функція	Ймовірність переходу	Коефіцієнти щільності ймовірності
1.	(1, 2)	W_{12}	P_1	$k=k_1$; $\theta=\theta_1$
2.	(2, 3)	W_{13}	P_2	$k=k_1$; $\theta=\theta_1$
3.	(1, 4)	W_{14}	$P_3=1-P_1-P_2$	$k=k_1$; $\theta=\theta_1$
4.	(2, 4)	W_{24}	P_4	$k=k_2$; $\theta=\theta_2$
5.	(2, 4)	W_{23}	P_5	$k=k_1$; $\theta=\theta_1$
6.	(3, 4)	W_{34}	P_8	$k=k_1$; $\theta=\theta_1$
7.	(4, 5)	W_{45}	P_{11}	$k=k_1$; $\theta=\theta_1$
8.	(3, 5)	W_{35}	P_9	$k=k_2$; $\theta=\theta_2$
9.	(2, 5)	W_{25}	P_6	$k=k_2$; $\theta=\theta_2$
10.	(4, 1)	W_{41}	$P_{12}=1-P_{11}$	$k=k_3$; $\theta=\theta_3$
11.	(3, 1)	W_{31}	$P_{10}=1-P_8-P_9$	$k=k_3$; $\theta=\theta_3$
12.	(3, 2)	W_{32}	$P_7=1-P_4-P_5-P_6$	$k=k_3$; $\theta=\theta_3$

Виконуючи функцію підстановки:

$$p_{\bar{b}}^{\bar{a}}(t)=P_{a_1}\int_{-\infty}^{\infty}e^{isx}\frac{x^{k_{b_1}-1}\cdot e^{\frac{-x}{\theta_{b_1}}}}{\theta_{b_1}^{k_{b_1}}\cdot\Gamma\left(k_{b_1}\right)}\mathrm{d}x\times P_{a_2}\int_{-\infty}^{\infty}e^{isx}\frac{x^{k_{b_2}-1}\cdot e^{\frac{-x}{\theta_{b_2}}}}{\theta_{b_2}^{k_{b_2}}\cdot\Gamma\left(k_{b_2}\right)}(x)\mathrm{d}x\times\ldots\times P_{a_n}\int_{-\infty}^{\infty}e^{isx}\frac{x^{k_{b_n}-1}\cdot e^{\frac{-x}{\theta_{b_n}}}}{\theta_{b_n}^{k_{b_n}}\cdot\Gamma\left(k_{b_n}\right)}(x)\mathrm{d}x, \quad (13)$$

отримуємо, що результуючий вираз розрахунку еквівалентних передавальних функцій можна описати як:

$$W_E(t)=\frac{\left(p_{(1,2)}^{(1,6)}(t)+p_{(1,2)}^{(2,9)}(t)+p_{(1,1)}^{(3,1)}(t)+p_{(1,2,1)}^{(1,4,1)}(t)+p_{(1,1,2)}^{(1,5,9)}(t)+p_{(1,1,1)}^{(2,8,11)}(t)+p_{(1,1,1,1)}^{(1,5,8,11)}(t)\right)}{1-\left(p_{(1,3)}^{(1,7)}(t)+p_{(1,3)}^{(1,10)}(t)+p_{(1,3)}^{(1,12)}(t)+p_{(1,2,3)}^{(1,4,12)}(t)+p_{(1,1,3)}^{(1,5,10)}(t)+p_{(1,1,3)}^{(2,8,12)}(t)+p_{(1,1,1,3)}^{(1,5,8,12)}(t)\right)}. \quad (14)$$

Щільність розподілу ймовірностей часу виконання алгоритму обфускації програмного коду:

$$\phi(x)=\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-isx}W_E(t)\mathrm{d}s=$$

$$=\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-itx}\frac{\left(p_{(1,2)}^{(1,6)}(t)+p_{(1,2)}^{(2,9)}(t)+p_{(1,1)}^{(3,1)}(t)+p_{(1,2,1)}^{(1,4,1)}(t)+p_{(1,1,2)}^{(1,5,9)}(t)+p_{(1,1,1)}^{(2,8,11)}(t)+p_{(1,1,1,1)}^{(1,5,8,11)}(t)\right)}{1-\left(p_{(1,3)}^{(1,7)}(t)+p_{(1,3)}^{(1,10)}(t)+p_{(1,3)}^{(1,12)}(t)+p_{(1,2,3)}^{(1,4,12)}(t)+p_{(1,1,3)}^{(1,5,10)}(t)+p_{(1,1,3)}^{(2,8,12)}(t)+p_{(1,1,1,3)}^{(1,5,8,12)}(t)\right)}\mathrm{d}t. \quad (15)$$

Мережа може бути побудована так, що якщо в деякому стані *i* можливий початок однієї з кількох наступних операцій, то ймовірності запуску p_{ij} в будь-якій з цих операцій утворюють повну групу несумісних подій. У такому випадку ймовірність виконання всієї мережі від витоку до стоку дорівнює 1.

Щоб показати, що всі вузли задовольняють цій умові, розрахуємо ймовірність виконання всього процесу:

$$p_E=\left.\frac{W_E(s)}{M_E(s)}\right|_{s=0}=W_E(0)=1, \quad (16)$$

де $M_E(s)$ – твірна функція, при цьому,

$$M_E(s)\big|_{s=0} = \int_{-\infty}^{\infty} e^{sx} f_E(x)\mathrm{d}x\Big|_{s=0} = \int_{-\infty}^{\infty} f_E(x)\mathrm{d}x = 1, \tag{17}$$

де $f_E(x)$ – функція щільності розподілу.

На рис. 2 показаний графік щільності розподілу часу виконання всього процесу обфускації програмного модуля з урахуванням варіацій значень змінних k та θ. Інтегрувавши щільність розподілу ймовірностей, одержимо функцію розподілу, графік якої відображений на рис. 3.

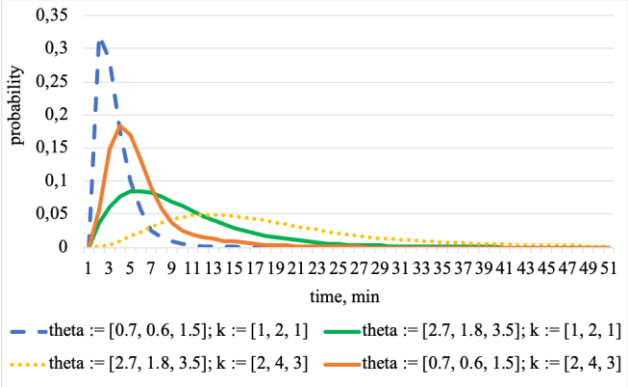


Рисунок 2 – Графіки щільності розподілу часу виконання процесу обфускації програмних модулів при використанні GERT-мережі

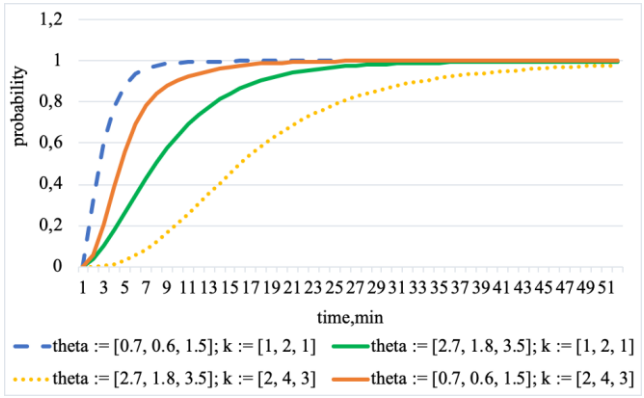


Рисунок 3 – Графіки функцій розподілу процесу обфускації програмних модулів при використанні GERT-мережі

Розроблена модель обфускації програмних модулів складається з 5 вузлів. Розглянемо поведінку системи при зміні кількості вузлів.

Розроблені GERT-мережі процесів обфускації програмних модулів зі зміненою кількістю вузлів представлені на рис. 4, 5.

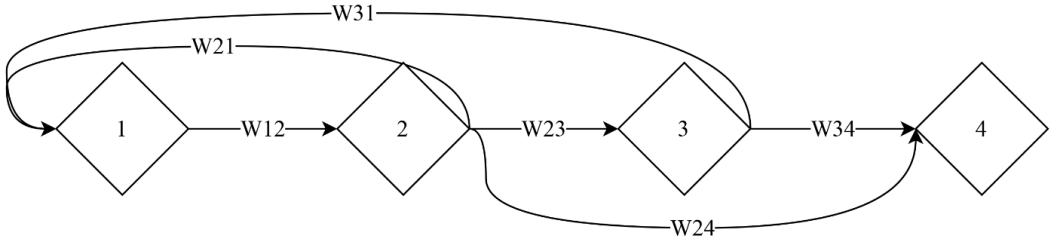


Рисунок 4 – GERT-мережа процесів обфускації програмних модулів зі зменшеною кількістю операторів функцій обфускації

При зміні кількості вузлів враховувалися такі фактори:

– ступінь зв’язності вузлів нового процесу порівнянний з оригінальним;

– зміна складності процесу призводить до зміни кількості елементів масиву коефіцієнтів k, при цьому значення першого і останнього елементів масиву k ідентичні оригінальним;

– зміна складності процесу призводить до зміни кількості елементів масиву коефіцієнтів θ, при цьому значення першого і останнього елементів масиву θ ідентичні оригінальним.

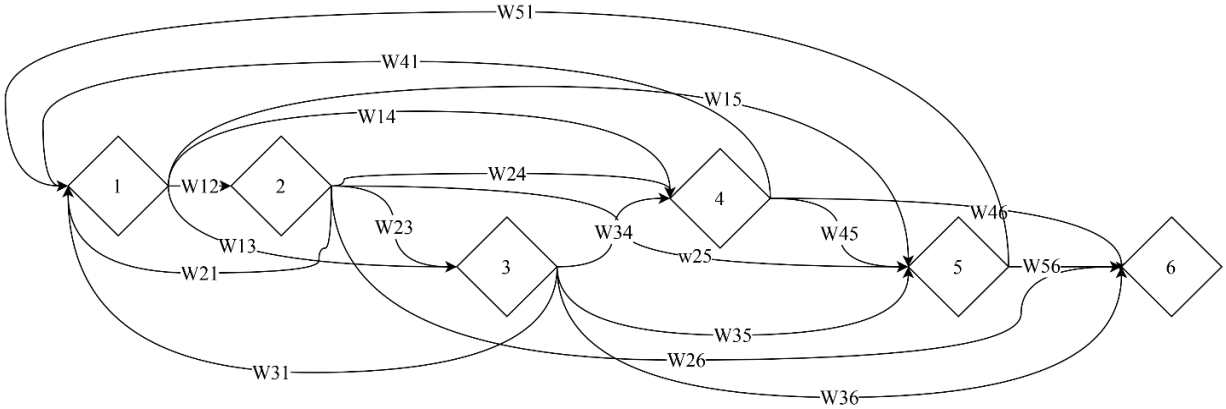


Рисунок 5 – GERT-мережа процесів обфускації програмних модулів з доданою додаткової функцією обфускації

Сформовані таблиці характеристик гілок розглянутих в GERT-моделі гілок і параметрів розподілу представлені в табл. 4, 5.

Таблиця 4 – Характеристики переходів між станами GERT-мережі процесів обфускації програмних модулів зі зменшеною кількістю оперованих функцій обфускації

№ з/п	Гілка	*W*-функція	Ймовірність переходу	Коефіцієнти щільності ймовірності (11)
1.	(1, 2)	W_{12}	P_1	$k=k_1$; $\theta=\theta_1$
2.	(2, 3)	W_{13}	P_2	$k=k_1$; $\theta=\theta_1$
3.	(3, 4)	W_{34}	P_3	$k=k_2$; $\theta=\theta_2$
4.	(2, 4)	W_{23}	P_4	$k=k_3$; $\theta=\theta_3$
5.	(2, 1)	W_{41}	$P_5=1-P_2-P_4$	$k=k_3$; $\theta=\theta_3$
6.	(3, 1)	W_{31}	$P_6=1-P_3$	$k=k_3$; $\theta=\theta_3$

Таблиця 5 – Характеристики переходів між станами GERT-мережі процесів обфускації програмних модулів з доданою додаткової оперується функцією обфускації

№ з/п	Гілка	*W*-функція	Імовірність переходу	Коефіцієнти щільності ймовірності (11)
1.	(1, 2)	W_{12}	P_1	$k=k_1$; $\theta=\theta_1$
2.	(2, 3)	W_{13}	P_2	$k=k_1$; $\theta=\theta_1$
3.	(3, 4)	W_{34}	P_3	$k=k_1$; $\theta=\theta_1$
4.	(4, 5)	W_{45}	P_4	$k=k_1$; $\theta=\theta_1$
5.	(5, 6)	W_{56}	P_5	$k=k_2$; $\theta=\theta_2$
6.	(2, 6)	W_{26}	P_6	$k=k_2$; $\theta=\theta_2$
7.	(3, 6)	W_{36}	P_7	$k=k_2$; $\theta=\theta_2$
8.	(4, 6)	W_{46}	P_8	$k=k_2$; $\theta=\theta_2$
9.	(1, 5)	W_{15}	P_9	$k=k_3$; $\theta=\theta_3$
10.	(2, 5)	W_{25}	P_{10}	$k=k_3$; $\theta=\theta_3$
11.	(3, 5)	W_{35}	P_{11}	$k=k_3$; $\theta=\theta_3$
12.	(1, 4)	W_{14}	P_{12}	$k=k_3$; $\theta=\theta_3$
13.	(2, 4)	W_{24}	P_{13}	$k=k_3$; $\theta=\theta_3$
14.	(1, 3)	W_{13}	$P_{14}=1-P_1-P_{12}-P_9$	$k=k_3$; $\theta=\theta_3$
15.	(2, 1)	W_{21}	$P_{15}=1-P_6-P_2-P_{13}-P_{10}$	$k=k_4$; $\theta=\theta_4$
16.	(3, 1)	W_{31}	$P_{16}=1-P_3-P_7-P_{11}$	$k=k_4$; $\theta=\theta_4$
17.	(4, 1)	W_{41}	$P_{17}=1-P_4-P_8$	$k=k_4$; $\theta=\theta_4$
18.	(5, 1)	W_{51}	$P_{18}=1-P_5$	$k=k_4$; $\theta=\theta_4$

Результуючі вираження розрахунку еквівалентних передавальних функцій можна описати як:

$$W_{E-1}(t)=\frac{p_{(1,2)}^{(1,4)}(t)+p_{(1,1,2)}^{(1,2,3)}(t)}{1-\left(p_{(1,3)}^{(1,5)}(t)+p_{(1,1,3)}^{(1,2,6)}(t)\right)}, \tag{18}$$

$$W_{E+1}(t)=\begin{pmatrix} p_{(3,2)}^{(9,5)}(t)+p_{(3,2)}^{(12,8)}(t)+p_{(3,2)}^{(14,7)}(t)+p_{(1,1,2)}^{(1,2,7)}(t)+ \\ +p_{(1,1,3,2)}^{(1,2,11,5)}(t)+p_{(1,3,2)}^{(1,10,5)}(t)+p_{(3,3,2)}^{(14,11,5)}(t)+ \\ +p_{(3,1,2)}^{(12,4,5)}(t)+p_{(1,1,1,2)}^{(1,2,3,8)}(t)+p_{(3,1,2)}^{(14,3,8)}(t)+ \\ +p_{(1,3,2)}^{(1,13,8)}(t)+p_{(1,2)}^{(1,6)}(t)+p_{(1,1,1,1,2)}^{(1,2,3,4,5)}(t)+ \\ +p_{(1,3,1,2)}^{(1,13,4,5)}(t)+p_{(3,1,1,2)}^{(14,3,4,5)}(t) \end{pmatrix} \Bigg/ 1- \begin{pmatrix} p_{(1,4)}^{(1,15)}(t)+p_{(1,1,4)}^{(1,2,16)}(t)+p_{(1,1,1,4)}^{(1,2,3,17)}(t)+ \\ +p_{(1,1,1,1,4)}^{(12,3,4,18)}(t)+p_{(3,4)}^{(14,16)}(t)+p_{(1,3,4)}^{(1,3,17)}(t)+ \\ +p_{(3,1,4)}^{(14,3,17)}(t)+p_{(3,4)}^{(12,17)}(t)+p_{(3,4)}^{(9,18)}(t)+ \\ +p_{(1,3,4)}^{(1,10,18)}(t)+p_{(3,3,4)}^{(14,11,18)}(t)+p_{(3,1,4)}^{(12,4,18)}(t)+ \\ p_{(1,1,3,4)}^{(1,2,11,18)}(t)+p_{(1,3,1,4)}^{(1,13,4,18)}(t)+p_{(3,1,1,4)}^{(14,3,4,18)}(t) \end{pmatrix}. \tag{19}$$

Для кожного нового процесу була вирахувана ймовірність виконання всього процесу p_E, що дорівнює 1. Також були розраховані значення математичного сподівання і дисперсії, що представлені в табл. 6.

Таблиця 6 – Характеристики щільності ймовірності – математичне сподівання і дисперсія

№ з/п	Тип	μ	σ^2
1.	Оригінальний θ=[2.7, 1.8, 3.5]; *k*=[1, 2, 1]	8.8	39.51
2.	Доданий вузол θ=[2.7, 1.8, 1.8, 3.5]; *k*=[1, 2, 2, 1]	11.11	44.68
3.	Видалений вузол θ=[2.7, 3.5]; *k*=[1, 1]	8.08	34.88

На основі отриманих еквівалентних передавальних функцій було побудовано графіки щільності розподілу часу виконання всього процесу обфускації і деобфускації програмного модуля з урахуванням варіацій значень змінних *k* та θ. Дані графіки, а також відповідні графіки функції розподілу, представлені на рис. 6, 7.

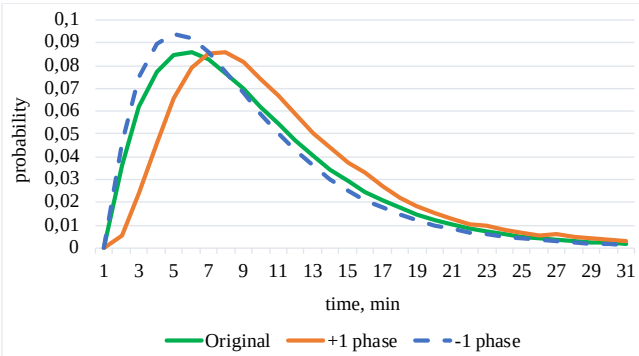


Рисунок 6 – Графіки щільності розподілу часу виконання процесу обфускації програмних модулів при використанні GERT-мережі

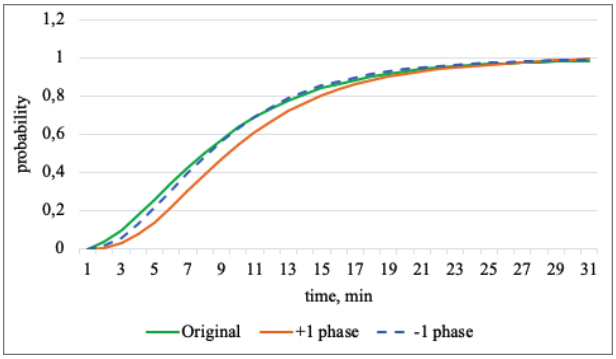


Рисунок 7 – Графіки функцій розподілу процесу обфускації програмних модулів при використанні GERT-мережі

Для дослідження якості розроблених методів обфускації програмних модулів були досліджені основні метрики якості коду для формування та можливості їх вдосконалення. Це дозволило кількісно оцінити ступінь обфускованості програмних продуктів.

В ході аналізу існуючих метрик якості коду та їх класифікації, запропоновано наступні вдосконалення для впровадження метрик коду:

1) Доповненням метрики середнього значення є метрика, що спрямована на визначення рівня вкладеності кожного модуля. У зв'язку з тим, що більшість обфускаторів коду для введення мертвого і невикористаного коду вводять додаткові цикли і умовні оператори, то підвищений рівень вкладеності модуля може свідчити про обфускацію коду.

2) Вдосконалення Fitzpatrick-метрики – врахування викликів операторів циклу (D). Таким чином, формула буде мати такий вигляд (A – кількість присвоювань, B – кількість викликів, C – кількість перевірок):

$$ABC = \sqrt{A^2 + B^2 + C^2 + D^2} \quad (20)$$

До умовних операторів слід відносити такі конструкції: 1) Оператор if – класичний умовний оператор, який використовується в даній метриці; 2) Тернарний оператор; 3) Оператори порівняння; 4) Операції фільтрації в сучасних механізмах обробки даних.

3) Розвиток метрики цикломатичної складності – оцінка ступеня взаємодії між модулями.

Для аналізу декомпільованих застосунків було розроблено програмну модель, яка для набору проектів робить зведену таблицю з інформацією щодо основних метрик, а також показує інформацію про частоти основних груп операторів. Основний крок алгоритму розробленого застосунку – отримання синтаксичного дерева. Отримання метрик вузла являє собою лінійний алгоритм, основне завдання якого – виклик функцій визначення конкретної метрики. Так, в дослідженні беруть участь метрики Спена, метрика рівня вкладеності, кількісні метрики груп операторів, метрика кількості рядків коду (LoC), FitzPatrik-метрика.

У рамках дослідження було сформовано інформацію про показники метрик з декомпільованим кодом та обфускованим декомпільованим у порівнянні з вихідним кодом. Результати наведені у табл. 7.

Таблиця 7 – Показники метрик з необфускованим декомпільованим кодом та обфускованим декомпільованим у порівнянні з вихідним кодом

Метрика	**Необфускований декомпільований код**	**Обфускований декомпільований код**
Фізична кількість строк коду	На 20% менш, ніж у вихідному коді	У 4 рази більш, ніж у вихідному коді (ВК)
Максимальна вкладеність	(як у вихідному коді)	На 20% більш, ніж у ВК
Значення Спену	На 10% менш, ніж у ВК	У 2 рази менш, ніж у ВК
Кількість `IF` блоків	(як у вихідному коді)	У 8 разів більш, ніж у ВК
Кількість `ELSE` блоків та тернарних операторів	На 20% менш, ніж у вихідному коді	У 7 разів більш, ніж у вихідному коді
Кількість `SWITCH` блоків	(як у вихідному коді)	У 5 разів більш, ніж у ВК
Кількість `GOTO` переходів	0 (як у вихідному коді)	PLoC * 0.11
Кількість `WHILE` операторів	У 2 рази менш, ніж у ВК	У 8 разів більш, ніж у ВК
Кількість `Do While` операторів	(як у вихідному коді)	PLoC * 0.01
Значення метрики FitzPatrik	На 10% менш, ніж у ВК	У 2 рази більш, ніж у ВК

У **четвертому розділі** представлено результати дослідження і алгоритми ліцензійної безпеки на основі водяних знаків для захисту авторських прав.

Розроблено комплекс математичних моделей системи безпеки програмного забезпечення. В основу математичного моделювання покладено підхід мережевого GERT-моделювання. В результаті розроблено математичні моделі системи безпеки програмного забезпечення з використанням операцій безпечного переходу та кодування ліцензійних ідентифікаторів.

Розроблено мережеву GERT-модель процесу ліцензійної безпеки програмного забезпечення, графічне зображення якої представлено на рис. 8.

У представленій мережі вузли графа інтерпретуються станами програмного продукту в процесі забезпечення ліцензійної безпеки, а гілки графа – ймовірнісно-часовими характеристиками переходів між станами. Зокрема, гілки (0,1), (0,2), (0,3), (0,4) описують процес взяття *i*-го біта ліцензійного ключа. Гілки (1,5), (2,5) відображають процес обробки двох бітів ліцензійного ключа. Гілка (5,0) відображає повернення на початковий стан; це може статися з двох причин: у разі обробки бітів пройшли або процес обробки бітів ще не повністю закінчився (наприклад, був оброблений тільки 0-ий біт і необхідно чекати обробку 1-го біта). Гілка (5,9) характеризує інформування системи про те, що група бітів успішно пройшла валідацію. Гілка (5,0) описує повернення на початковий стан. Це може статися у зв’язку з невалідністю вхідних даних. Гілка (3,9) характеризує інформування системи про те, що група бітів успішно пройшла валідацію.

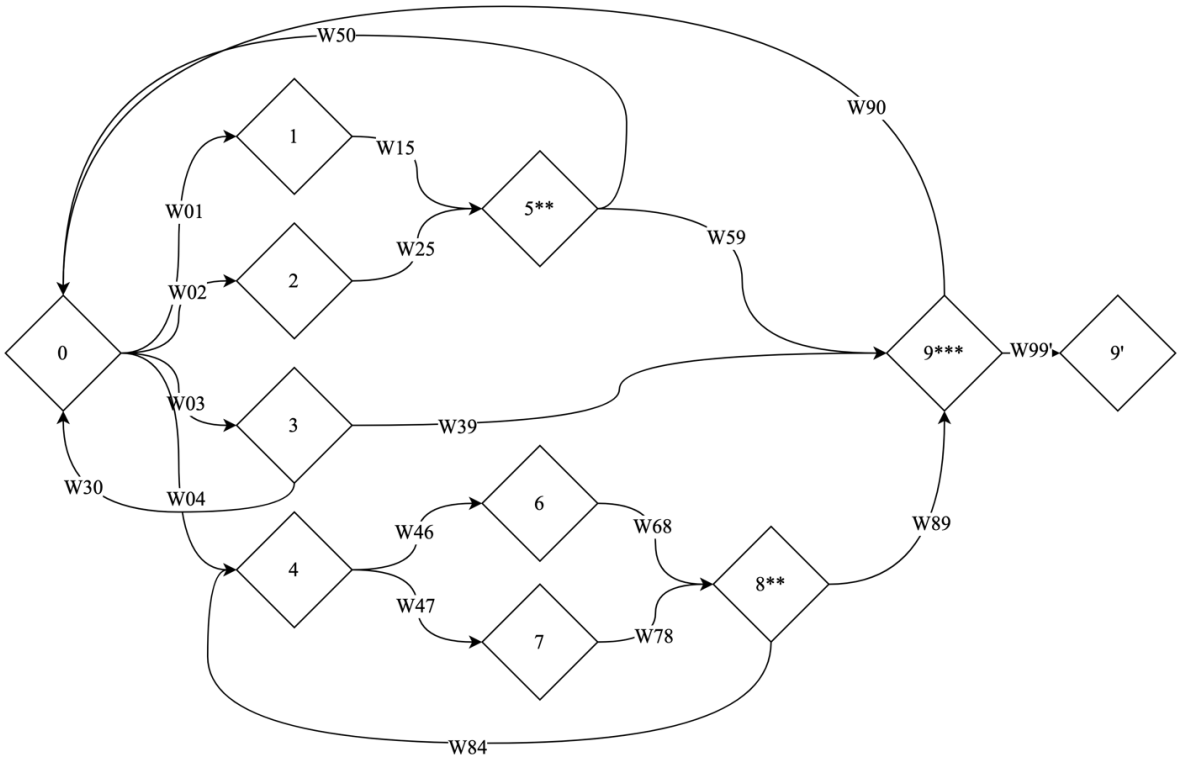


Рисунок 8 – GERT-мережу процесу ліцензійної безпеки програмного забезпечення

Гілки (4,6), (4,7), (6,8), (7,8) описують процес обробки старшого біта ліцензійного ключа, в ході якого відбувається виконання корисного коду і переходи на відповідні стани. Гілка (8,4) відображає повернення на початковий стан; це може

статися з двох причин: у разі обробки бітів пройшли або процес обробки бітів ще не повністю закінчився (наприклад, був оброблений тільки 0-ий біт і необхідно чекати обробку 1-го біта). Гілка (8,9) характеризує інформування системи про те, що група бітів успішно пройшла валідацію. Гілка (9,0) повернення на початковий стан. Це може статися з двох причин: у разі обробки бітів пройшли або процес обробки бітів ще не повністю закінчився. Гілка (9,9') описує успішність перевірки всіх бітів та перехід на стан штатної роботи програмного продукту.

Еквівалентна *W*-функція часу виконання алгоритмів обфускації програмного коду дорівнює:

$$W_E(s) = \frac{W'_E(s)}{1 - W''_E(s)}; \tag{21}$$

$$W'_E(s) = W_{01}W_{15}W_{59}W_{99`} + W_{02}W_{25}W_{59}W_{99`} + W_{03}W_{39}W_{99`} + W_{04}W_{46}W_{68}W_{89}W_{99`} + W_{04}W_{47}W_{78}W_{89}W_{99`}; \tag{22}$$

$$W''_E(s) = W_{01}W_{15}W_{50} + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{84} + W_{04}W_{47}W_{78}W_{84} + \\ + W_{01}W_{15}W_{59}W_{90} + W_{02}W_{25}W_{59}W_{90} + W_{03}W_{39}W_{90} + W_{04}W_{46}W_{68}W_{89}W_{90} + W_{04}W_{47}W_{78}W_{89}W_{90}. \tag{23}$$

Сформована таблиця характеристик гілок розглянутих в GERT-моделі гілок і параметрів розподілу представлена в табл. 8.

Використовуючи описаний в розділі 3 вираз, який уособлює добуток *W*-функцій, що описують успішне і неуспішне виконання алгоритмів, отримаємо результуючий вираз розрахунку еквівалентних передаточних функцій:

$$W_E(t) = \frac{p_{1,2,3,3}^{1,5,7,17} + p_{1,2,3,3}^{2,6,7,17} + p_{1,3,3}^{3,9,17} + p_{1,1,2,3,3}^{4,11,13,15,17} + p_{1,1,2,3,3}^{4,12,14,15,17}}{1 - \left(p_{1,2,4}^{1,5,8} + p_{1,2,4}^{2,6,8} + p_{1,4}^{3,10} + p_{1,1,2,4}^{4,11,13,16} + p_{1,1,2,4}^{4,12,14,16} + p_{1,2,3,4}^{1,5,7,18} + p_{1,2,3,4}^{2,6,7,18} + p_{1,3,4}^{3,9,18} + p_{1,1,2,3,4}^{4,11,13,15,18} + p_{1,1,2,3,4}^{4,12,14,15,18} \right)}.$$

Таблиця 8 - Характеристики переходів між станами GERT-мережі процесу ліцензійної безпеки програмного забезпечення

№ з/п	Гілка	*W*-функція	Ймовірність переходу	Коефіцієнти щільності ймовірності (11)
1.	(0, 1)	W_{01}	P_1	$k=k_1$; $\theta=\theta_1$
2.	(0, 2)	W_{02}	P_2	$k=k_1$; $\theta=\theta_1$
3.	(0, 3)	W_{03}	P_3	$k=k_1$; $\theta=\theta_1$
4.	(0, 4)	W_{04}	$P_4=1-P_1-P_2-P_3$	$k=k_1$; $\theta=\theta_1$
5.	(1, 5)	W_{15}	P_5	$k=k_2$; $\theta=\theta_2$
6.	(2, 5)	W_{25}	P_6	$k=k_2$; $\theta=\theta_2$
7.	(5, 9)	W_{59}	P_7	$k=k_3$; $\theta=\theta_3$
8.	(5, 0)	W_{50}	$P_8=1-P_7$	$k=k_4$; $\theta=\theta_4$
9.	(3, 9)	W_{39}	P_9	$k=k_3$; $\theta=\theta_3$
10.	(3, 0)	W_{30}	$P_{10}=1-P_9$	$k=k_4$; $\theta=\theta_4$
11.	(4, 6)	W_{46}	P_{11}	$k=k_1$; $\theta=\theta_1$
12.	(4, 7)	W_{47}	$P_{12}=1-P_{11}$	$k=k_1$; $\theta=\theta_1$
13.	(6, 8)	W_{68}	P_{13}	$k=k_2$; $\theta=\theta_2$
14.	(7, 8)	W_{78}	P_{14}	$k=k_2$; $\theta=\theta_2$
15.	(8, 9)	W_{89}	P_{15}	$k=k_3$; $\theta=\theta_3$
16.	(8, 4)	W_{84}	$P_{16}=1-P_{15}$	$k=k_4$; $\theta=\theta_4$
17.	(9, 9`)	$W_{99`}$	P_{17}	$k=k_3$; $\theta=\theta_3$
18.	(9, 0)	W_{90}	$P_{18}=1-P_{17}$	$k=k_4$; $\theta=\theta_4$

Використовуючи щільність розподілу ймовірностей, отримуємо графік розподілу щільності ймовірності, відображений на рис. 9.

На рис. 9 показаний графік щільності розподілу часу виконання всього процесу забезпечення ліцензійного захисту програмного забезпечення при $k=[2,4,4,3]$ і $\theta=[2.7,1.8,1.8,3.5]$. Інтегрувавши щільність розподілу ймовірностей, одержимо функцію розподілу, графік якої відображений на рис. 10.

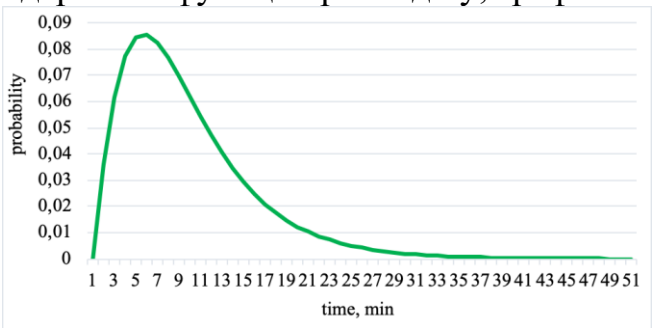


Рисунок 9 – Графік щільності розподілу часу виконання процесу забезпечення ліцензійної безпеки програмного продукту при використанні GERT-мережі

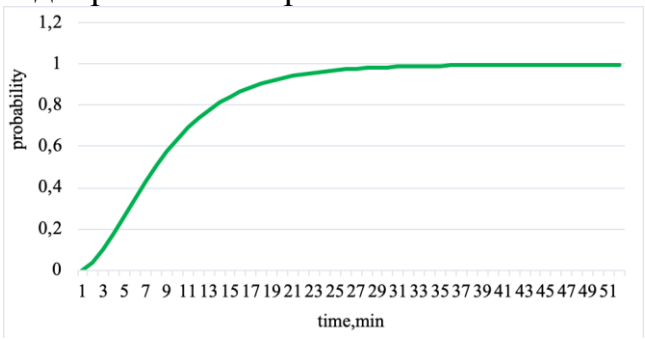


Рисунок 10 – Графік функцій розподілу процесу забезпечення ліцензійної безпеки програмного продукту при використанні GERT-мережі

Математичне сподівання і дисперсія отриманих функцій розраховуються відповідно до формул:

$$\mu = W_E(t)\mathrm{d}t\,|_{t=0} = 3.52, \tag{24}$$

$$\sigma^2 = W_E(t)\mathrm{d}^2t\,|_{t=0} - \mu^2 = 117.01. \tag{25}$$

У рамках дисертаційного дослідження наведено дослідження довжини ключа при використанні GERT-мереж шляхом нарощування (масштабування).

В рамках дослідження було прийнято використовувати 2 типи масштабування: горизонтальне (рис. 11) і вертикальне (рис. 12).

При горизонтальному масштабуванні частина ключа вклинюється в процес обробки конкретних бітів. Наприклад, з рис. 11 видно, що ми перейдемо в стан 5 в разі, коли не лише біти 0 і 1 валідні, а й коли біти 4-7 валідні (стан 1‘-4’). Характеристики переходів між станами GERT-мережі описані в табл. 9. Як видно з даної таблиці, розвинена GERT-мережа позбавлена гілки (0-5). У свою чергу, було додано ряд гілок, що з’єднуються з вершинами 0 і 5 таким же чином, як це робилося для зв’язку 0-ої вершини з вершинами 1-4 і вершини 9 з вершинами 3, 5, 8 відповідно.

Згідно до табл. 9, результуюча W-функція горизонтального масштабування має такий вигляд:

$$W_{EH}(s) = \frac{W'_{EH}(s)}{1 - W''_{EH}(s)}. \tag{26}$$

$$W'_{EH}(s) = W_{01}W_{11'}W_{1'5'}W_{5'5}W_{59}W_{99'} + W_{01}W_{12'}W_{2'5'}W_{5'5}W_{59}W_{99'} + W_{01}W_{13'}W_{3'5}W_{59}W_{99'} + W_{01}W_{14'}W_{4'6'}W_{6'8'}W_{8'5}W_{59}W_{99'} + \\ + W_{01}W_{14'}W_{4'7'}W_{7'8'}W_{8'5}W_{59}W_{99'} + W_{02}W_{25}W_{59}W_{99'} + W_{03}W_{39}W_{99'} + W_{04}W_{46}W_{68}W_{89}W_{99'} + W_{04}W_{47}W_{78}W_{89}W_{99'} \tag{27}$$

Таблиця 9 – Характеристики переходів між станами GERT-мережі процесу ліцензійної безпеки програмного забезпечення при горизонтальному масштабуванні

№ з/п	Гілка	*W*-функція	Ймовірність переходу	Коефіцієнти щільності ймовірності
1.	(0, 1), (1, 1')	W_{01}, $W_{11'}$	P_1	$k=k_1$; $\theta=\theta_1$
2.	(0, 2) , (1, 2')	W_{02}, $W_{12'}$	P_2	$k=k_1$; $\theta=\theta_1$
3.	(0, 3) , (1, 3')	W_{03}, $W_{13'}$	P_3	$k=k_1$; $\theta=\theta_1$
4.	(0, 4) , (1, 4')	W_{04}, $W_{41'}$	$P_4=1-P_1-P_2-P_3$	$k=k_1$; $\theta=\theta_1$
5.	(1`, 5`)	$W_{1'5'}$	P_5	$k=k_2$; $\theta=\theta_2$
6.	(2, 5) , (2', 5')	W_{25}, $W_{2'5'}$	P_6	$k=k_2$; $\theta=\theta_2$
7.	(5, 9) , (5', 5)	W_{59}, $W_{5'5}$	P_7	$k=k_3$; $\theta=\theta_3$
8.	(5, 0) , (5`, 1)	W_{50}, $W_{5`1}$	$P_8=1-P_7$	$k=k_4$; $\theta=\theta_4$
9.	(3, 9) , (3', 5)	W_{39}, $W_{3'5}$	P_9	$k=k_3$; $\theta=\theta_3$
10.	(3, 0) , (3', 1)	W_{30}, $W_{3'1}$	$P_{10}=1-P_9$	$k=k_4$; $\theta=\theta_4$
11.	(4, 6) , (4', 6')	W_{46}, $W_{4'6'}$	P_{11}	$k=k_1$; $\theta=\theta_1$
12.	(4, 7) , (4', 7')	W_{47}, $W_{4'7'}$	$P_{12}=1-P_{11}$	$k=k_1$; $\theta=\theta_1$
13.	(6, 8) , (6', 8')	W_{68}, $W_{6'8'}$	P_{13}	$k=k_2$; $\theta=\theta_2$
14.	(7, 8) , (7', 8')	W_{78}, $W_{7'8'}$	P_{14}	$k=k_2$; $\theta=\theta_2$
15.	(8, 9) , (8', 5)	W_{89}, $W_{8'5}$	P_{15}	$k=k_3$; $\theta=\theta_3$
16.	(8, 4) , (8', 4')	W_{84}, $W_{8'4'}$	$P_{16}=1-P_{15}$	$k=k_4$; $\theta=\theta_4$
17.	(9, 9`)	$W_{99`}$	P_{17}	$k=k_3$; $\theta=\theta_3$
18.	(9, 0)	W_{90}	$P_{18}=1-P_{17}$	$k=k_4$; $\theta=\theta_4$

$$
\begin{aligned}
W''_{EH}(s) = {} & W_{01}W_{11'}W_{1'5'}W_{5'1} + W_{01}W_{12'}W_{2'5'}W_{5'1} + W_{01}W_{11'}W_{1'5'}W_{5'5}W_{50} + W_{01}W_{12'}W_{2'5'}W_{5'5}W_{50} + \\
& + W_{01}W_{11'}W_{1'5'}W_{5'5}W_{59}W_{90} + W_{01}W_{12'}W_{2'5'}W_{5'5}W_{59}W_{90} + W_{01}W_{14'}W_{4'6'}W_{6'8'}W_{8'4'} + W_{01}W_{13'}W_{3'1} + \\
& + W_{01}W_{13'}W_{3'5}W_{50} + W_{01}W_{13'}W_{3'5}W_{59}W_{90} + W_{01}W_{14'}W_{4'7'}W_{7'8'}W_{8'4'} + W_{01}W_{14'}W_{4'6'}W_{6'8'}W_{8'5}W_{50} + \\
& + W_{01}W_{14'}W_{4'7'}W_{7'8'}W_{8'5}W_{50} + W_{01}W_{14'}W_{4'6'}W_{6'8'}W_{8'5}W_{59}W_{90} + W_{01}W_{14'}W_{4'7'}W_{7'8'}W_{8'5}W_{59}W_{90} + \\
& + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{84} + W_{04}W_{47}W_{78}W_{84} + W_{02}W_{25}W_{59}W_{90} + W_{03}W_{39}W_{90} + \\
& + W_{04}W_{46}W_{68}W_{89}W_{90} + W_{04}W_{47}W_{78}W_{89}W_{99}
\end{aligned}
\tag{28}
$$

Результуючий вираз розрахунку еквівалентних передавальних функцій матиме такий вигляд:

$$
W_{EH}(t) = \frac{\begin{array}{c} p_{1,1,2,3,3,3}^{1,1,5,7,7,17} + p_{1,1,2,3,3,3}^{1,2,6,7,7,17} + p_{1,1,3,3,3}^{1,3,9,7,17} + p_{1,1,1,2,3,3,3}^{1,4,11,13,15,7,17} + p_{1,1,1,2,3,3,3}^{1,4,13,14,15,7,17} + \\ + p_{1,2,3,3}^{2,2,7,17} + p_{1,3,3}^{3,9,17} + p_{1,1,2,3,3}^{4,11,13,15,17} + p_{1,1,2,3,3}^{4,12,14,15,17} \end{array}}{1 - \left(\begin{array}{c} p_{1,1,2,4}^{1,1,5,8} + p_{1,1,2,4}^{1,2,6,8} + p_{1,1,4}^{1,3,10} + p_{1,1,1,2,4}^{1,4,11,13,16} + p_{1,1,1,2,4}^{1,4,12,14,16} + p_{1,1,2,3,4}^{1,1,5,7,8} + p_{1,1,2,3,4}^{1,2,6,7,8} + p_{1,1,3,4}^{1,3,9,8} + p_{1,1,1,2,3,4}^{1,4,11,13,15,8} + \\ + p_{1,1,1,2,3,4}^{1,4,12,14,15,8} + p_{1,1,2,3,3,4}^{1,1,5,7,7,18} + p_{1,1,2,3,3,4}^{1,2,6,7,7,18} + p_{1,1,3,3,4}^{1,3,9,7,18} + p_{1,1,1,2,3,3,4}^{1,4,11,13,15,7,18} + p_{1,1,1,2,3,3,4}^{1,4,12,14,15,7,18} + \\ + p_{1,2,4}^{2,6,8} + p_{1,4}^{3,10} + p_{1,1,2,4}^{4,11,13,16} + p_{1,1,2,4}^{4,12,14,16} + p_{1,2,3,4}^{2,6,7,18} + p_{1,3,4}^{3,9,18} + p_{1,1,2,3,4}^{4,11,13,15,18} + p_{1,1,2,3,4}^{4,12,14,15,18} \end{array} \right)}
\tag{29}
$$

При вертикальному масштабуванні відбувається модифікація переходів на перші стани (наприклад, (0-1), (0-2)) шляхом додавання нових гілок. При цьому, загальна ймовірність переходу зі стану 0 зменшується пропорційно довжині ключа. В даному випадку, складність графа залишається подібною початковій. Характеристики переходів між станами GERT-мережі описані в табл. 10.

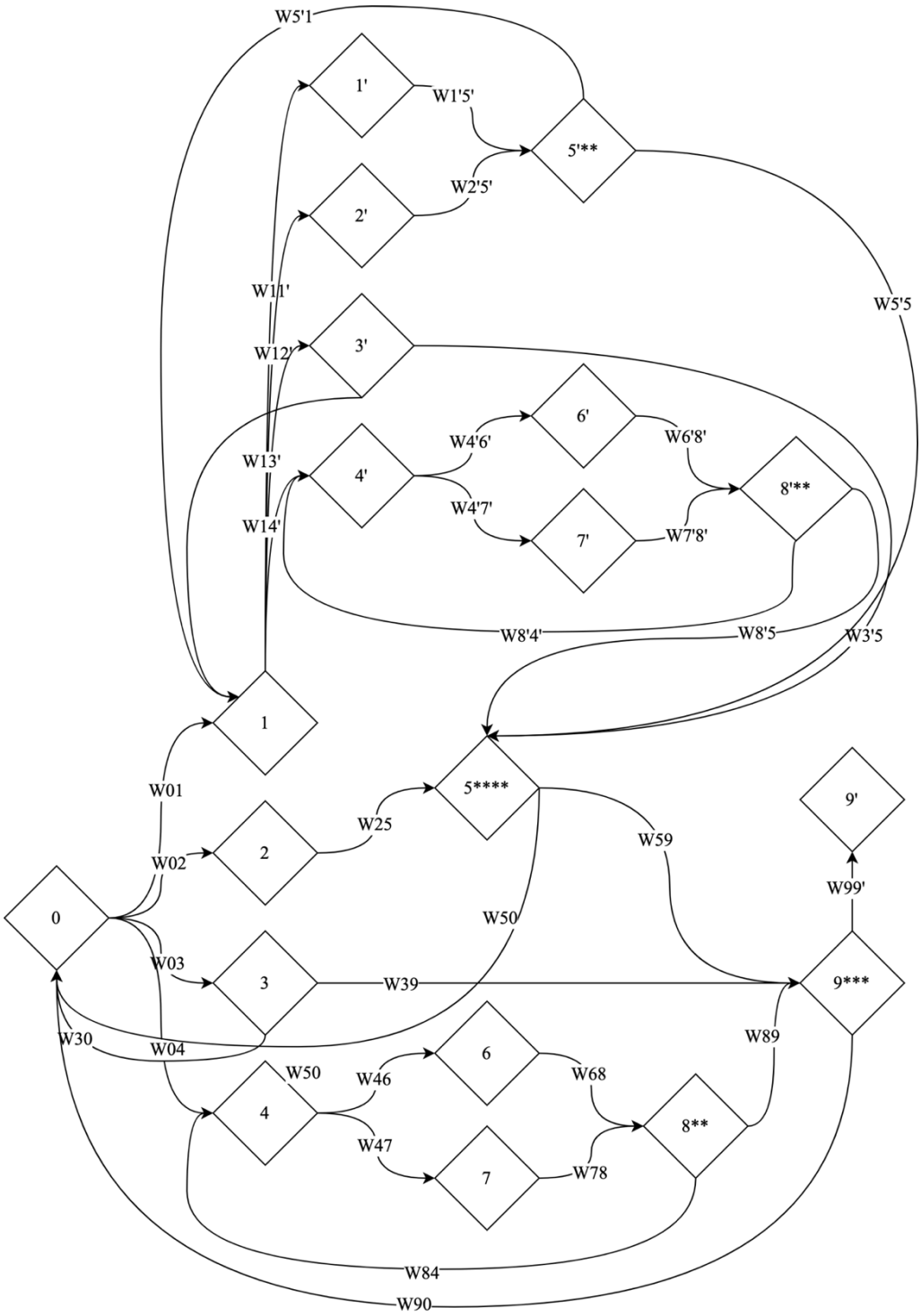


Рисунок 11 – GERT-мережа процесу ліцензійної безпеки програмного забезпечення при горизонтальному масштабуванні

Згідно табл. 10, результуюча *W*-функція вертикального масштабування має такий вигляд:

$$W_{EV}(s) = \frac{W'_{EV}(s)}{1 - W''_{EV}(s)} \tag{30}$$

$$W'_{EV}(s) = W_{01}W_{15}W_{59}W_{99'} + W_{02}W_{25}W_{59}W_{99'} + W_{03}W_{39}W_{99'} + W_{04}W_{46}W_{68}W_{89}W_{99'} + W_{04}W_{47}W_{78}W_{89}W_{99'} + \\ + W_{01'}W_{1'5'}W_{5'9}W_{99'} + W_{02'}W_{2'5'}W_{5'9}W_{99'} + W_{03'}W_{3'9}W_{99'} + W_{04'}W_{4'6'}W_{6'8'}W_{8'9}W_{99'} + W_{04'}W_{4'7'}W_{7'8'}W_{8'9}W_{99'} \tag{31}$$

$$W''_{EV}(s) = W_{01}W_{15}W_{50} + W_{02}W_{25}W_{50} + W_{03}W_{30} + W_{04}W_{46}W_{68}W_{84} + W_{04}W_{47}W_{78}W_{84} + W_{01}W_{15}W_{59}W_{90} + W_{02}W_{25}W_{59}W_{90} + \\ + W_{03}W_{39}W_{90} + W_{04}W_{46}W_{68}W_{89}W_{90} + W_{04}W_{47}W_{78}W_{89}W_{99'} + W_{01'}W_{1'5'}W_{5'0} + W_{02'}W_{2'5'}W_{5'0} + W_{03'}W_{3'0} + W_{04'}W_{4'6'}W_{6'8'}W_{8'4'} + \\ + W_{04'}W_{4'7'}W_{7'8'}W_{8'4'} + W_{01'}W_{1'5'}W_{5'9}W_{90} + W_{02'}W_{2'5'}W_{5'9}W_{90} + W_{03'}W_{3'9}W_{90} + W_{04'}W_{4'6'}W_{6'8'}W_{8'9}W_{90} + W_{04'}W_{4'7'}W_{7'8'}W_{8'9}W_{99'} \quad (32)$$

Таблиця 10 – Характеристики переходів між станами GERT-мережі процесу ліцензійної безпеки програмного забезпечення при вертикальному масштабуванні

№ з/п	Гілка	*W*-функція	Ймовірність переходу	Коефіцієнти щільності ймовірності
1.	(0, 1), (0, 1`)	W_{01}, $W_{01'}$	$P_1=P_{1`}$	$k=k_1$; $\theta=\theta_1$
2.	(0, 2) , (0, 2`)	W_{02}, $W_{02'}$	$P_2=P_{2`}$	$k=k_1$; $\theta=\theta_1$
3.	(0, 3) , (0, 3`)	W_{03}, $W_{03'}$	$P_3=P_{3`}$	$k=k_1$; $\theta=\theta_1$
4.	(0, 4) , (0, 4`)	W_{04}, $W_{04'}$	$P_4=P_{1`}=(1-P_1-P_2-P_3-P_{1`}-P_{2`}-P_{3`})/2$	$k=k_1$; $\theta=\theta_1$
5.	(1, 5) , (1`, 5`)	W_{15}, $W_{1`5'}$	P_5	$k=k_2$; $\theta=\theta_2$
6.	(2, 5) , (2`, 5`)	W_{25}, $W_{2`5'}$	P_6	$k=k_2$; $\theta=\theta_2$
7.	(5, 9) , (5`, 9`)	W_{59}, $W_{5`9}$	P_7	$k=k_3$; $\theta=\theta_3$
8.	(5, 0) , (5`, 0)	W_{50}, $W_{5`0}$	$P_8=1-P_7$	$k=k_4$; $\theta=\theta_4$
9.	(3, 9) , (3`, 9`)	W_{39}, $W_{3`9'}$	P_9	$k=k_3$; $\theta=\theta_3$
10.	(3, 0) , (3`, 0)	W_{30}, $W_{3'0}$	$P_{10}=1-P_9$	$k=k_4$; $\theta=\theta_4$
11.	(4, 6) , (4`, 6`)	W_{46}, $W_{4`6'}$	P_{11}	$k=k_1$; $\theta=\theta_1$
12.	(4, 7) , (4`, 7`)	W_{47}, $W_{4`7'}$	$P_{12}=1-P_{11}$	$k=k_1$; $\theta=\theta_1$
13.	(6, 8) , (6`, 8`)	W_{68}, $W_{6`8'}$	P_{13}	$k=k_2$; $\theta=\theta_2$
14.	(7, 8) , (7`, 8`)	W_{78}, $W_{7`8'}$	P_{14}	$k=k_2$; $\theta=\theta_2$
15.	(8, 9) , (8`, 9`)	W_{89}, $W_{8`9}$	P_{15}	$k=k_3$; $\theta=\theta_3$
16.	(8, 4) , (8`, 4`)	W_{84}, $W_{8`4'}$	$P_{16}=1-P_{15}$	$k=k_4$; $\theta=\theta_4$
17.	(9, 9`)	$W_{99`}$	P_{17}	$k=k_3$; $\theta=\theta_3$
18.	(9, 0)	W_{90}	$P_{18}=1-P_{17}$	$k=k_4$; $\theta=\theta_4$

Результуючий вираз розрахунку еквівалентних передавальних функцій матиме такий вигляд:

$$W_{EV}(t) = \frac{\left(\begin{array}{c} p_{1,2,3,3}^{1,5,7,17} + p_{1,2,3,3}^{2,6,7,17} + p_{1,3,3}^{3,9,17} + p_{1,1,2,3,3}^{4,11,13,15,17} + p_{1,1,2,3,3}^{4,12,14,15,17} + \\ + p_{1,2,3,3}^{1',5,7,17} + p_{1,2,3,3}^{2',2,7,17} + p_{1,3,3}^{3',9,17} + p_{1,1,2,3,3}^{4',11,13,15,17} + p_{1,1,2,3,3}^{4',12,14,15,17} \end{array}\right)}{1 - \left(\begin{array}{c} p_{1,2,4}^{1,5,8} + p_{1,2,4}^{2,6,8} + p_{1,4}^{3,10} + p_{1,1,2,4}^{4,11,13,16} + p_{1,1,2,4}^{4,12,14,16} + p_{1,2,3,4}^{1,5,7,18} + p_{1,2,3,4}^{2,6,7,18} + p_{1,3,4}^{3,9,18} + p_{1,1,2,3,4}^{4,11,13,15,18} \\ + p_{1,1,2,3,4}^{4,12,14,15,18} + p_{1,2,4}^{1',5,8} + p_{1,2,4}^{2',6,8} + p_{1,4}^{3',10} + p_{1,1,2,4}^{4',11,13,16} + p_{1,1,2,4}^{4',12,14,16} + p_{1,2,3,4}^{1',5,7,18} + \\ + p_{1,2,3,4}^{2',6,7,18} + p_{1,3,4}^{3',9,18} + p_{1,1,2,3,4}^{4',11,13,15,18} + p_{1,1,2,3,4}^{4',12,14,15,18} \end{array}\right)} \quad (33)$$

Використовуючи щільність розподілу ймовірностей (11), отримуємо графік розподілу щільності ймовірності для розробленої початкової GERT-мережі процесу ліцензійної безпеки програмного забезпечення і її модифікацій (при горизонтальному і вертикальному масштабуванні). Результати наведені на рис. 13, 14.

Наведені графіки щільності розподілу ймовірностей для GERT-мереж процесу ліцензійної безпеки програмного забезпечення на рис. 13 використовують такі коефіцієнти щільності ймовірності: $k=[2,4,4,3]$ та $\theta=[2.7,1.8,1.8,3.5]$. Інтегрувавши щільність розподілу ймовірностей, отримаємо функції розподілу, графіки яких відображені на рис. 14.

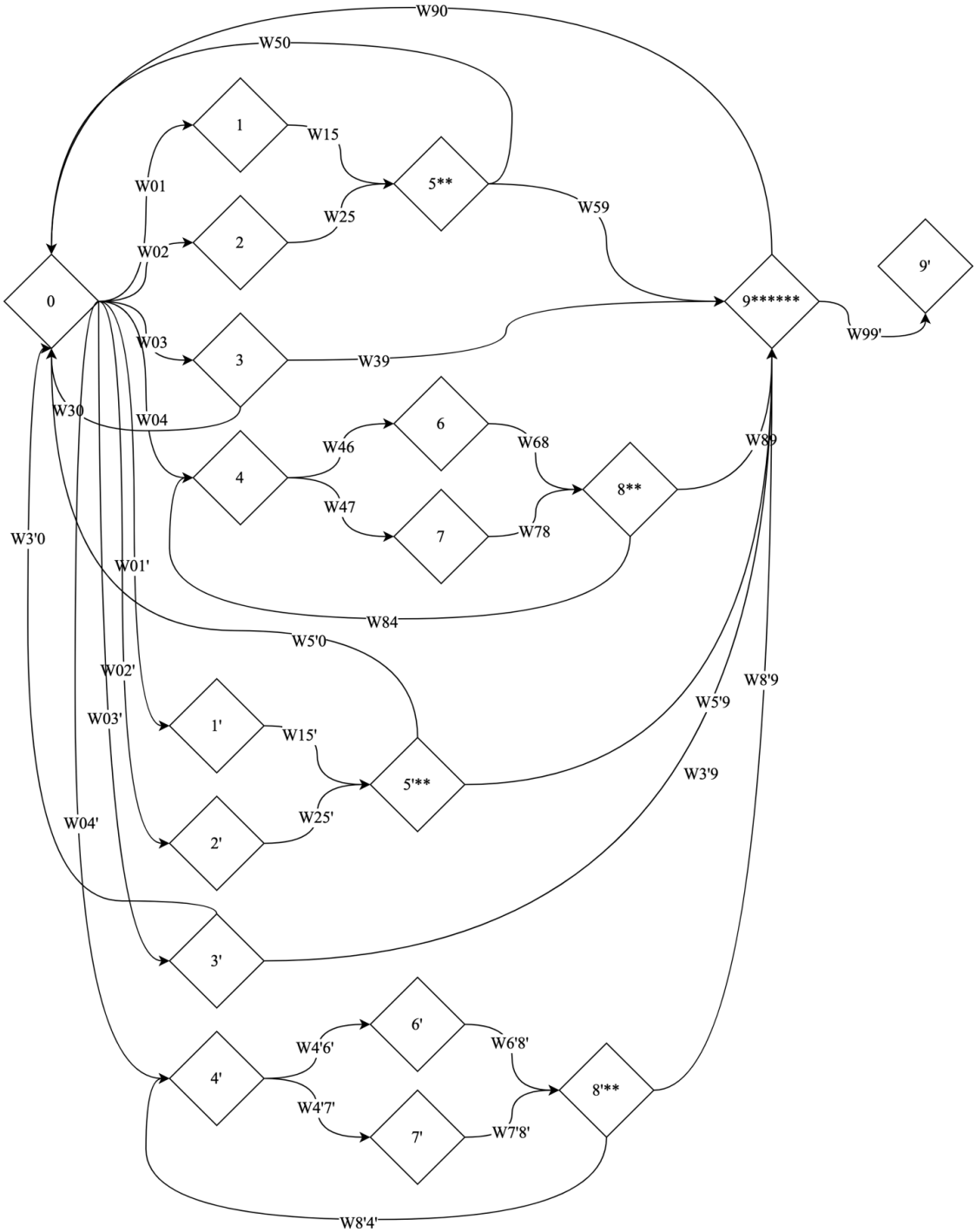


Рисунок 12 – GERT-мережа процесу ліцензійної безпеки програмного забезпечення при вертикальному масштабуванні

Математичне сподівання і дисперсія отриманих функцій:

– для вертикального масштабування:

$$\mu = W_E(t)\mathrm{d}t\,|_{t=0} = 3.71, \quad \sigma^2 = W_E(t)\mathrm{d}^2t\,|_{t=0} - \mu^2 = 130.01.$$

– для горизонтального масштабування:

$$\mu = W_E(t)\mathrm{d}t\,|_{t=0} = 12.11, \quad \sigma^2 = W_E(t)\mathrm{d}^2t\,|_{t=0} - \mu^2 = 230.01.$$

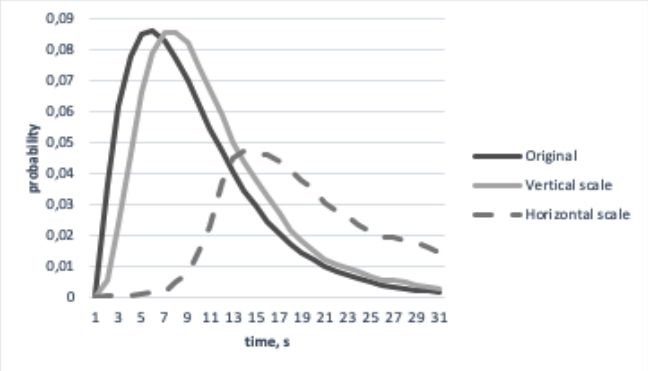


Рисунок 13 – Порівняльні графіки щільності розподілу часу виконання процесу забезпечення ліцензійної безпеки програмного продукту при використанні GERT-мережі

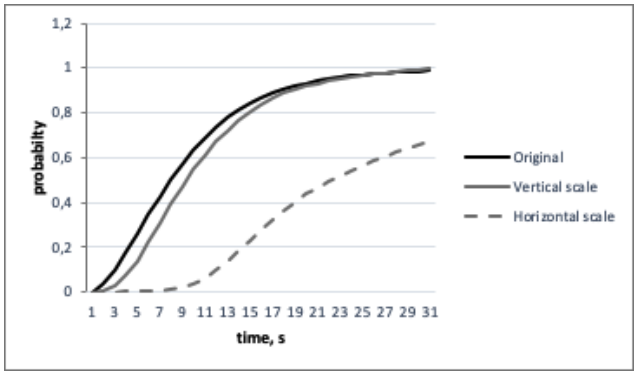


Рисунок 14 – Порівняльні графіки функцій розподілу процесу забезпечення ліцензійної безпеки програмного продукту при використанні GERT-мережі

У п’ятому розділі розроблено метод формування цифрового ідентифікатора програмного забезпечення, відмінною особливістю якого є використання індивідуальних даних про комп’ютерні системи кінцевого користувача для однозначної ідентифікації приналежності, на які ліцензійне програмне забезпечення встановлюється в процесі формування ліцензійного цифрового ідентифікатора. Це дає можливість підвищити безпеку програмного забезпечення шляхом захисту від неліцензійного копіювання.

В рамках дослідження сформовано модель процесу формування цифрового ідентифікатора програмного забезпечення. Її структурна схема представлена на рис. 15.

Як видно з рисунка, основними складовими розроблюваної моделі є:

– менеджер ліцензії. Цей модуль є розпорядником всіх інших компонентів моделі. У його завдання входить порівняння унікальних даних персонального комп’ютера, на якому запущена копія продукту, з тими, що закодовані всередині продукту; виклик модуля контролю цілісності; генерація ліцензійного ключа на основі даних про цільову комп’ютерної системи;

– модуль контролю цілісності. Мета даного компонента: підпис ліцензійного ключа (CRC-сума); підпис програмного продукту цифровим сертифікатом (X.509) кінцевого користувача; періодична перевірка цілісності модулів системи безпеки на основі ліцензійного ключа в процесі експлуатації програмного продукту. Такі перевірки є додатковим бар’єром на шляху зловмисника, якщо він спробує модифікувати або фальсифікувати компоненти системи;

– модуль ідентифікації. У завдання цього компонента входить отримання даних, однозначно ідентифікуючих персональний комп’ютер. Як дані можуть виступати унікальні серійні номери апаратних комплектуючих ПК (наприклад, жорсткого диска, процесора, вбудованої камери, Bluetooth, Wi-Fi модулів і т.д.), дати створення BIOS, ідентифікатор операційної системи (для ОС Windows - серійний номер);

– інтеграція ліцензійного ключа з програмним продуктом. Даний модуль отримує інформацію про можливий додатковий код, що виконується в режимі Runtime для індивідуального програмного продукту;

– обфускація вихідного коду виконуючого модуля.

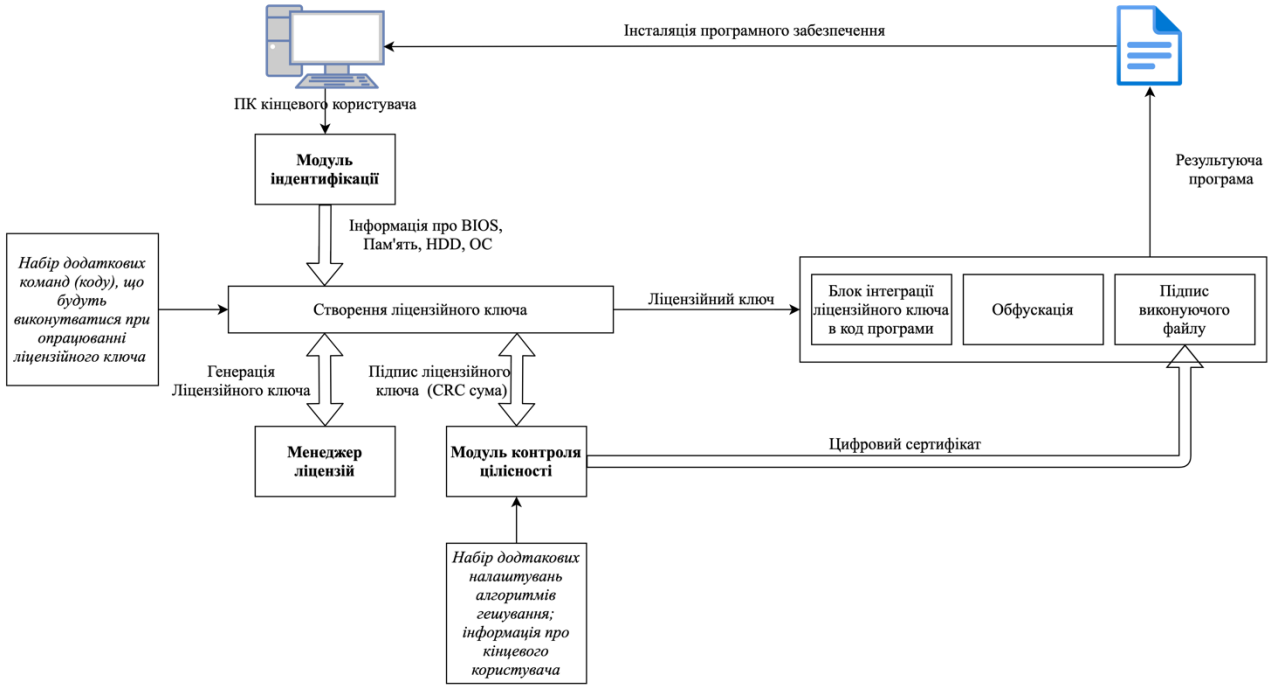


Рисунок 15 – Структурна схема процесу формування цифрового ідентифікатора програмного забезпечення

Відповідно до процедур формування ліцензійного цифрового ідентифікатора необхідно виконати наступні кроки:

1. *Формування шаблону програмного коду, на основі якого генерується ліцензійний ключ*: запис в глобальну змінну MODE значення необхідного режиму роботи програми (список варіюється для індивідуального програмного продукту), від якого залежить функціонал (FULL – вся функціональність доступна; DEMO – доступна лише «базова» функціональність); порівняння типу продукту, версії, порівнянності поточного програмного продукту з встановленим типом операційної системи. У разі розбіжності програми можливо кілька варіантів розвитку ситуації залежно від вимог постачальника програмного забезпечення: виведення відповідного повідомлення користувачу і завершення програми; зменшення функціональності до мінімально можливої (наприклад, демо-версія); зменшення функціональності до тільки-для-читання (без можливості збереження проміжних результатів).

2. *Отримання інформації про компоненти комп'ютерної системи кінцевого користувача*. Ця функція виконується шляхом створення програми-утиліти, яка запускається на системі кінцевого користувача і відправляє по захищеному каналу необхідну інформацію, щоб уникнути перехоплення трафіку і його аналізу / зміни на сервер. Отримана інформація буде зберігатися для подальших дій. Програма-утиліта знаходиться на захищеному носії, доступ до якого є тільки у авторизованого співробітника (даний варіант не виключає ймовірність проникнення зловмисників, але в контексті пропонованої реалізації, дії даного чинника ігноруються).

Інформація, яка надходить на сервер про компоненти системи кінцевого користувача, має таку структуру:

– накопичувачі, наприклад: «WDC WD5000AAKX-001CA0; PCI\VEN_8086; DEV_1C02; SUBSYS_844D1043; REV_05\3»;

– процесори, наприклад: «Intel(R) Pentium(R) CPU G620 @ 2.60GHz; ACPI_HAL\PNP0C08\0»;

– ОС, наприклад: «6.3.9600 N/A Build 9600; 00261-80443-28329-AA407»;

– BIOS, наприклад: «American Megatrends Inc. 0409, 8/26/2011».

3. *Формування результуючого ліцензійного ключа*. Для виконання даної функції в якості вхідних даних використовуються: інформація про систему кінцевого користувача; шаблон програмного коду, який буде виконуватися; розмір CRC-суми для валідації ліцензійного ключа.

Узагальнений алгоритм процесу формування цифрового ідентифікатора ПЗ має наступний вигляд:

1. Формування програмного коду (для .Net – файл машиного коду на асемблері (для ОС Windows), для JVM – java-файл), який буде виконуватися на системі кінцевого користувача з урахуванням отриманої інформації та наданого шаблону.

2. Отриманий код трансформується в відповідні машинні / байт коди.

3. Для кожного байта отриманого коду виконуються наступні дії:

– пошук даного байта в текстовому поданні отриманої інформації;

– пошук даного байта в бінарному представленні отриманої інформації;

– якщо знайдений відповідний код в текстовому поданні, то в кінець формується ліцензійного ключа дописується 2 байта: «0» і «код пристрою», якщо в бінарному представленні - то 2 байта: «1» і «код пристрою», потім 2 байта, що відображають позицію знайденого символу. У разі, якщо код не був знайдений - 2 байта «00», а потім 2 байта, що відображають hex-уявлення зазначеного байта (напр., Символ «A» має код «65», що в hex-поданні - 41 - буде записано два байта «4» і «1»).

В кінець отриманого ліцензійного ключа додається 2 байта, що характеризують ступінь CRC-суми, наприклад «32». Далі йде CRC-сума в текстовому hex-поданні (для CRC32 – 8 символів). У процесі формування цифрового ідентифікатора ПЗ можна використовувати такі види пристроїв: 1 - накопичувачі; 2 - процесори; 3 - ОС; 4 - BIOS. У цьому випадку на виході отримуємо згенерований ліцензійний ключ, який буде вбудовуватися в програмний продукт. Вихідний програмний продукт проходить 3 фази перед надходженням до кінцевого користувача:

1. Отриманий ліцензійний ключ вбудовується в сегмент даних програми. При цьому сегмент даних повинен бути сформований таким чином, щоб ліцензійний ключ помістився у виділеному для цього блоці.

2. Обфускація, для зменшення ймовірності зламу алгоритму обробки ліцензійного ключа.

3. Підпис програмного продукту сертифікатом, щоб уникнути можливості редагування файлу з метою обійти алгоритм верифікації.

Далі отриманий програмний комплекс поставляється кінцевому користувачеві. При кожному запуску, програма виконує наступні дії:

1. Отримує інформацію про компоненти поточної системи.
2. Верифікує ліцензійний ключ на основі CRC-суми.
3. Формує машинні коди для виконання на основі поточного ліцензійного ключа і параметрів системи.
4. Виконує сформовані машинні команди. У разі, якщо ліцензійний ключ виявився невалідним або програма була запущена на іншому комп'ютері, то поточний програмний продукт аварійно завершить роботу.

Запропонувалось ліцензійний ключ вбудовувати в блок даних. Весь виконуваний файл підписується електронним цифровим підписом з метою можливості перевірки цілісності та автентичності.

На основі аналізу літератури, переваг і недоліків існуючих механізмів формування та верифікації ліцензійного ключа, була зроблена спроба розробки програмного комплексу, яка зменшить ймовірність злочинного впливу на ліцензійний ключ. На основі аналізу переваг існуючих комплексів, в першу чергу, була зроблена спроба захистити програмний комплекс від злочинного впливу шляхом обфускації коду, закликаної збільшити час аналізу алгоритму програми зловмисником і використовувати підписи виконуваного файлу, щоб уникнути модифікування.

Базовими стратегіями захисту ліцензійного ключа від тиражування є наступні:

– використання інформації про складові обчислювальної системи кінцевого користувача;
– ядро ліцензійного ключа являє собою функцію (програмний код), яка буде виконуватися на системі кінцевого користувача.

Процес формування ліцензійного ключа:

– з боку кінцевого користувача передається інформація про систему на сервер по захищеному каналу зв'язку (наприклад, протокол https);
– сервер підбирає програмний код, який буде виконуватися на стороні клієнта, в залежності від переданих параметрів (ОС, тип ліцензії);
– сервер кодує обраний програмний код з додаванням CRC32-суми;
– сервер відправляє кінцевому користувачеві ліцензійний ключ по захищеному каналу зв'язку;
– програма-клієнт кінцевого користувача зберігає закодований ключ, з постійним його декодуванням при кожному запуску.

Шостий розділ присвячено оцінці ефективності розроблених моделей та методів підвищення безпеки байт-код орієнтованого програмного забезпечення і оцінці достовірності результатів дисертаційного дослідження.

Для обґрунтування достовірності отриманих результатів та оцінки ефективності розроблених методів, було проведено імітаційне моделювання підсистем:

– захисту на основі обфускації програмного коду;
– захисту на основі формування ліцензійних ідентифікаторів;

– аналізу вимог до ймовірісно-часових показників програмного забезпечення.

У якості інструментарію для імітаційного моделювання використовувався пакет Mapple-2021.

Узагальнена структурна схема розробленої імітаційної моделі системи підвищення безпеки програмного забезпечення наведена на рис. 16.

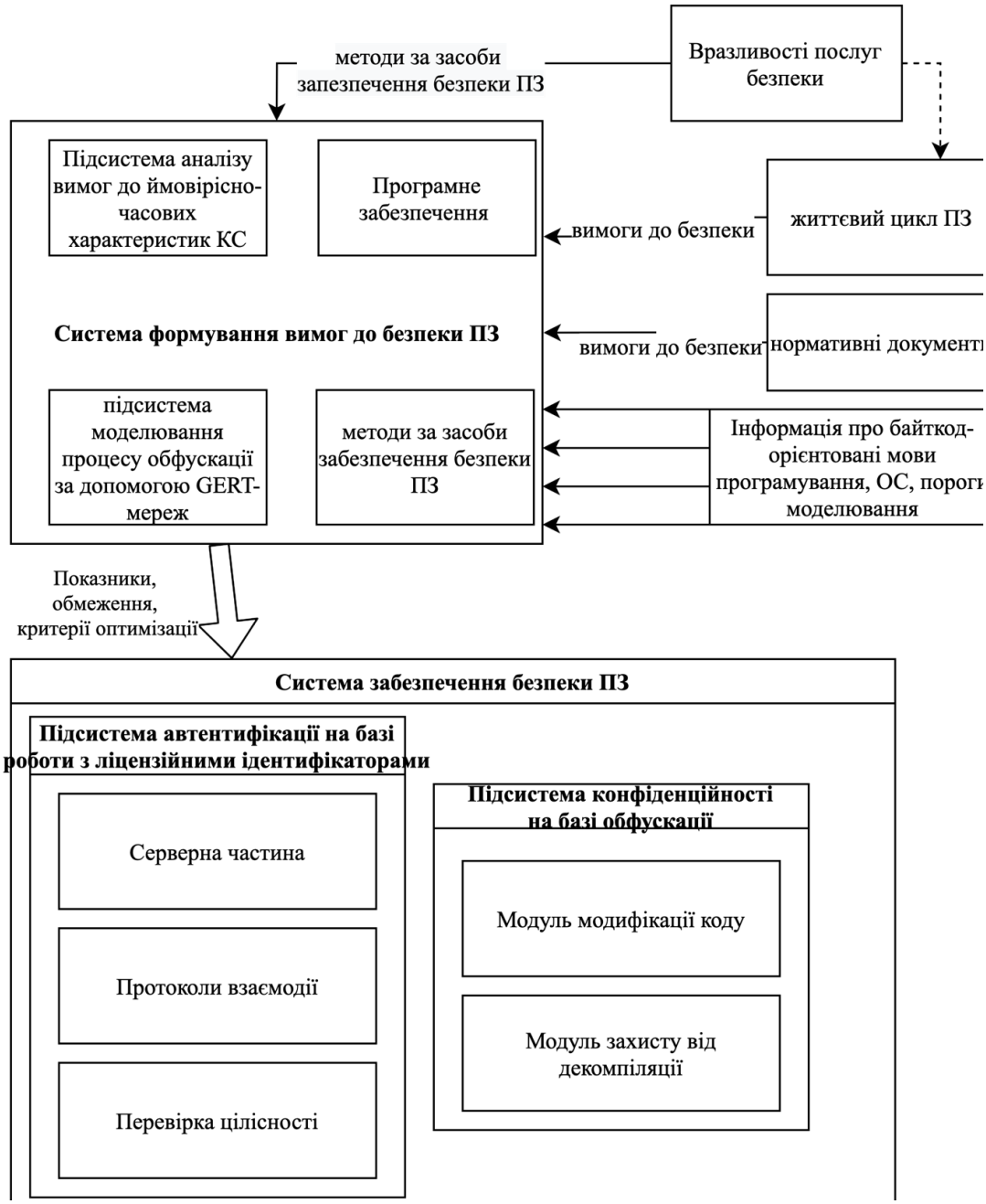


Рисунок 16 – Структурна схема імітаційної моделі систем формування вимог до безпеки та захисту програмного забезпечення

До складу програмного комплексу імітаційного моделювання входять дві системи, які виконують окремі функції:

– вироблення показників, обмежень і критеріїв оптимізації процесів обфускації коду та генерації ліцензійних ключів, а також вироблення відповідних керуючих команд;

– вироблення рішень про захист програмного забезпечення в процесі його експлуатації.

У процесі формування вимог до безпеки програмного забезпечення емулювалися і розглядалися різні показники і характеристики інформаційного обміну, типи системного програмного забезпечення, можливі варіанти архітектурної побудови програмного забезпечення. Зокрема розглядалися такі ситуації:

– кількість і типи використовуваних операційних систем (в даному випадку використовувалися наступні операційні системи: Windows 8, 10; Linux Ubuntu 16; MacOS X 11);

– кількість і типи використовуваних байт-код орієнтованих мов програмування (в даному випадку використовувалися наступні мови програмування: .Net 6, .Net Core 3, JVM 8, 11);

– зв'язність мережі, кількість функціональних вузлів і зв'язку між ними. Даний параметр задається у вигляді масиву вузлів, кожний з яких має список вузлів-«сусідів». Вузол представлений у вигляді набору наступних параметрів: назва вузла, тип ОС вузла, список вузлів-«сусідів». Даний показник впливає на модуль формування ліцензійного ключа при взаємодії роботи клієнтського ПЗ та серверу ліцензійного модулю.

З урахуванням вищевказаних факторів, в програмному комплексі здійснюється вибір показників, обмежень і критеріїв оптимізації процесів забезпечення безпеки програмного забезпечення. При цьому формуються команди для підсистеми захисту на основі систем обфускації програмного коду та формування ліцензійного ідентифікатора.

Проведено порівняльні дослідження з відомими методами, використовуючи методи і прийоми математичного та імітаційного моделювання.

Для підтвердження ефективності розробленого методу в порівнянні з існуючими, знайдемо співвідношення часу злому ліцензійного ключа: t_{MI} / t_{Mi}..

У рамках дослідження час злому ліцензійного ключа, що був створений завдяки розробленому методу підвищення безпеки програмного забезпечення в умовах використання байт-код орієнтованих мов програмування (МПБПЗ) був порівняний з існуючими методами:

– метод клієнт-серверної взаємодії при верифікації ліцензійного ключа (КСВ);

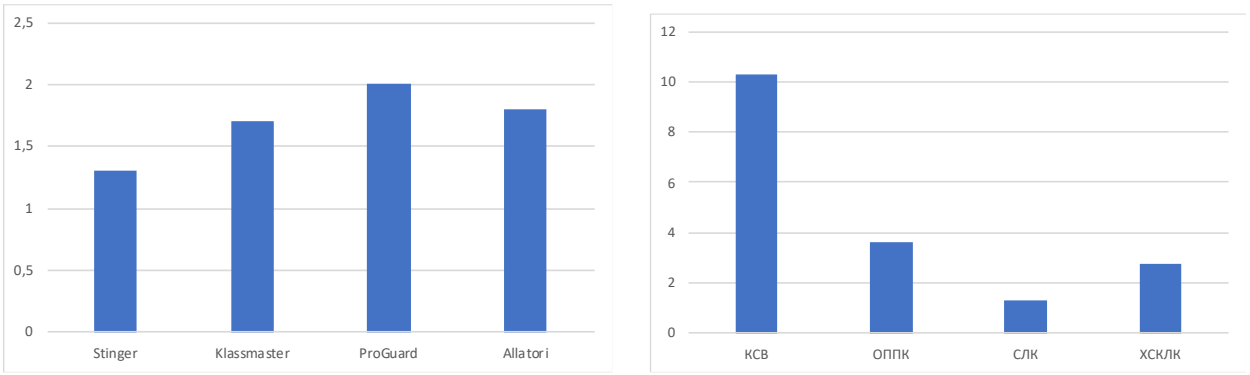
– метод обфускації коду і методу підпису / верифікації ліцензійного ключа за допомогою системи публічного / приватного ключів (ОППК);

– метод статичних ліцензійних ключів в залежності від наданої функціональності (СЛК);

– метод використання алгоритмів гешування (MD2, MD5, SHA1) і симетричного кодування ліцензійного ключа (ХСКЛК).

Дані експерименту наведені на рис. 17, 18. Як бачимо з рис. 18, мінімальна ефективність розробленого методу складає 1.4 рази (у порівнянні з існуючим

методом обфускації коду і методу підпису / верифікації ліцензійного ключа за допомогою системи публічного / приватного ключів).



а) б)

Рисунок 17 – Графік співвідношення середнього часу зламу програмного забезпечення, що використовує розроблений метод обфускації (а) та генерації ліцензійного ключа (б) до програмних застосунків, що використовують існуючі методи захисту програмного забезпечення

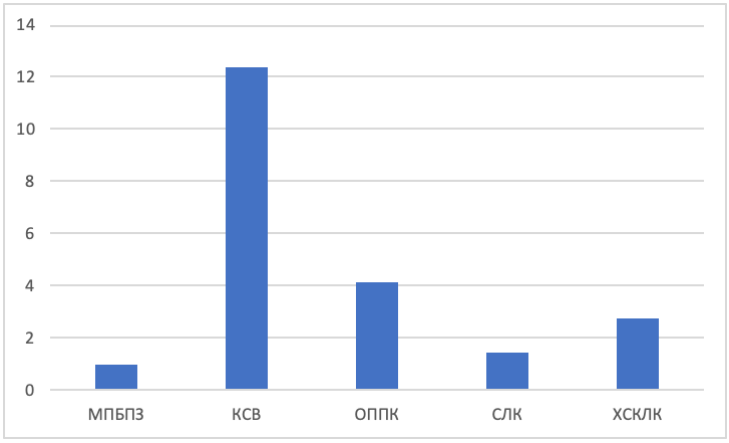


Рисунок 18 – Графік співвідношення середнього часу зламу програмного забезпечення розроблених методів до програмних застосунків, що використовують існуючі методи захисту програмного забезпечення

Для обґрунтування достовірності отриманих попередніх розділах результатів проведено імітаційне моделювання процесу підвищення безпеки програмного забезпечення шляхом поліпшення методу обфускації програмних модулів і генерації ліцензійного ключа.

Висунута в дисертаційній роботі гіпотеза про нормальний розподіл випадкової величини часу зламу ліцензійного ключа була перевірена за критерієм згоди χ^2 Пірсона.

Проведена перевірка довела правдоподібність гіпотези про те, що величина часу зламу розподілена за нормальним законом.

Отримані оцінки $w(\vec{\xi})^{(i)}$ математичного сподівання і $\hat{D}_{w(\vec{\xi})^{(i)}}$ дисперсії ($\hat{\sigma}_{w(\vec{\xi})^{(i)}}$ середнього відхилення) випадкової величини $w(\vec{\xi})^{(i)}$.

Скориставшись виразом для розрахунку довірчої ймовірності відхилення відносної частоти від постійної ймовірності в незалежних випробуваннях, отримане

в результаті експерименту значення прогнозованого рівня квантування «не відхилене» від математичного сподівання $\hat{w}(\vec{\xi})^{(i)}$ більш ніж на 1.

Поведінка імітаційного моделювання показала, що для всіх досліджуваних видів даних довірча ймовірність того, що значення статистичної величини $w(\vec{\xi})$ «не відхилиться» від математичного сподівання $w(\vec{\xi})$ більш ніж на 1: $P \approx 0{,}98$.

Високий ступінь збігу результатів імітаційного та математичного моделювання підтверджує достовірність розробленого методу підвищення безпеки програмного забезпечення в умовах використання байт-код орієнтованих мов програмування.

У висновках сформульовані основні результати дисертаційного дослідження.

В додатках наведено документи, що підтверджують практичне значення і впровадження результатів дисертаційної роботи.

ВИСНОВКИ

В дисертаційній роботі вирішено науково-практичну проблему підвищення безпеки байт-код орієнтованого програмного забезпечення в умовах кібератак на основі розробки моделей та методів синтезу підсистем обфускації програмного коду та генерації ліцензійних ключів. Проведені в дисертаційній роботі дослідження, результати вирішення науково-технічної проблеми і окремих наукових завдань, а також результати порівняльного аналізу, дали можливість отримати наступні наукові та практичні результати.

1. Отримав подальший розвиток метод перевірки логіко-смислової подібності програм складної логічної структури, що відрізняється від відомих розпаралелюванням процесів, що обчислюються при порівнянні подібних елементів програмного коду, а також формуванням та використанням графу спільних обчислень в процесі пошуку подібних елементів коду, що в свою чергу зменшило складність процесу верифікації.

2. Розроблено GERT-модель процесу обфускації програмних модулів, що реалізує парадигми використання математичного апарату гамма-розподілу у якості ключового на всіх етапах моделювання процесу обфускації. Це дозволило досягти уніфікації моделі в умовах модифікації GERT-мережі. Результати дослідження показали, що для розробленої математичної моделі при додаванні ще одного процесу обфускації дисперсія часу виконання збільшується на 12%, а при його видаленні з системи – зменшується до 13%. Математичне сподівання часу виконання змінюється в геометричній прогресії – так, при видаленні вузла відбувається зменшення математичного сподівання на 9%, а при збільшенні на 1 вузол – збільшення математичного сподівання на 26%. Це показує незначність змін досліджуваних показників в умовах модифікації моделі і підтверджує гіпотезу про уніфікацію моделі в умовах використання математичного апарату гамма-розподілу як основного. Дані результати дають розробнику можливість спрогнозувати поведінку системи захисту програмних модулів з точки зору часу виконання. Це дозволяє зменшити час на прийняття рішення про доцільність використання процесу обфускації в умовах використання гнучких методологій.

3. Вдосконалено технологію обфускації програмних модулів, що відрізняється від відомих урахуванням варіативності типів лексем та ідентифікаторів. Це дозволило підвищити безпеку програмного забезпечення.

4. Розроблено критерій оцінки якості обфускації програмного забезпечення шляхом синтезу лінійної композиції часткових критеріїв метрик якості коду, що дозволило кількісно оцінити ступінь обфускованості програмних продуктів.

5. Розроблено модель безпечного переходу і кодування ліцензійних ідентифікаторів на основі математичного апарату GERT-мереж з парадигмою гамма-розподілу, що дозволило підвищити точність результатів моделювання. Дана логіка впроваджується в залежності від ідентифікаційного або серійного номера. Розроблено методологію масштабування розробленої математичної моделі. Показано доцільність використання кожного типу масштабування з урахуванням критичності часу виконання здійснення перевірки безпеки програмного забезпечення на основі ліцензійних ідентифікаторів. Так, при вертикальному масштабуванні, в зв'язку з використанням паралельного процесу обробки даних, час обробки даних практично не змінився. Це дає передумови використання даного типу масштабування при критичності часу виконання процесу, а також при необхідності істотного нарощування довжини ліцензійного ключа на слабких пристроях (наприклад, вбудованих пристроях та IoT). При горизонтальному масштабуванні, в зв'язку з використанням додаткових послідовних процесів обробки даних, час обробки значно збільшився (в 3.43 рази). Однак, введення додаткових послідовних дій збільшує час аналізу алгоритму поведінки. Це дає передумови використання даного типу масштабування при використанні прикладного програмного забезпечення, де час запуску програми не критичний.

6. Розроблено метод формування цифрового ідентифікатора програмного забезпечення, відмінною особливістю якого є використання індивідуальних даних про комп'ютерні системи кінцевого користувача для однозначної ідентифікації приналежності, на які ліцензійне програмне забезпечення встановлюється в процесі формування ліцензійного цифрового ідентифікатора. Це дає можливість підвищити безпеку програмного забезпечення шляхом захисту від неліцензійного копіювання. Запропоновано алгоритм функціонування системи і генерації ліцензійного ключа, адаптований до вхідних даних і можливих умов верифікації програмного забезпечення. Також, модель враховує можливість вбудовування довільного (заданого розробниками) коду в тіло ліцензійного ключа, який буде виконуватися при верифікації. Дані маніпуляції призвели до ускладнення аналізу і зламу ліцензійної складової програмного забезпечення. Це дозволило підвищити середній час зламу в 1.29 разів.

Проведено оцінку достовірності та ефективності запропонованих методів і моделей підвищення безпеки байт-код орієнтованого програмного забезпечення.

Практична значущість отриманих результатів полягає в наступному. Отримані результати підтверджуються їх застосуванням у діяльності компаній «Line Up», «Нікс Солюшенс ЛТД», Державного підприємства «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості», Державного підприємства «Харківський науково-дослідний інститут технологій машинобудування», а також використання у навчальному

процесі Національного технічного університету «Харківський політехнічний інститут». Використання результатів дисертаційної роботи підтверджене відповідними актами впровадження.

Наукове використання результатів, отриманих в дисертаційній роботі, можливе у рамках подальшого розвитку наукового напряму, який пов'язаний з розробкою та удосконаленням ефективних методологій забезпечення безпеки ПЗ.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Semenov S., Davydov V., Lipchanska O., Lipchanskyi M. Development of unified mathematical model of programming modules obfuscation process based on graphic evaluation and review method // Eastern-European Journal of Enterprise Technologies. 2020. № 3(2(105)). P. 6–16. (**SCOPUS**).

2. Давидов В. В., Гавриленко С. Ю., Прохорова Т. М. Дослідження методів побудови синтаксичних аналізаторів // Системи обробки інформації. 2015. № 11(136). С. 125-128.

3. Давидов В. В., Семенов С. Г., Зиков І. С. Дослідження ризиків моніторингу технічного стану об'єктів авіації // Системи озброєння і військова техніка. 2015. № 4(44). С. 108-110.

4. Давидов В. В., Мовчан А. В., Сидоренко И. И. Разработка системы формирования цифрового идентификатора программного обеспечения для защиты авторских прав // Системи обробки інформації. 2016. № 3. С. 15-17.

5. Давыдов В. В., Гребенюк Д. С. Комплекс процедур генерации лицензионного ключа для защиты авторских прав интеллектуальной собственности на программное обеспечение // Системи управління, навігації та зв'язку. 2017. № 1(41). С. 11-15.

6. Давидов В. В., Пасько Д. А., Молчанов Г. І. Управління необмеженою кількістю хмарних сховищ // Сучасні інформаційні технології. 2018. Том 2. №3. С. 49–53.

7. Давидов В. В., Гавриленко С. Ю., Челак В. В. Разработка системы фиксации аномальных состояний компьютера // Вісник Національного технічного університету "ХПІ": Серія: Інформатика та моделювання. 2018. № 42 (1318). С. 109 – 121.

8. Давидов В. В., Бульба С. С., Кучук Г. А. Метод розподілу ресурсів між композитними застосунками // Системи управління, навігації та зв'язку. 2018. Том 4 (50). С. 99-104.

9. Семенов С. Г., Давидов В. В., Волошин Д. Г., Гребенюк Д. С. Метод захисту модуля програмного забезпечення на основі процедури обфускації // Телекомунікаційні та інформаційні технології. 2019. № 4 (65). С. 71–80.

10. Davydov V., Hrebeniuk D. Development of the methods for resource reallocation in cloud computing systems // Innovative Technologies and Scientific Solutions for Industries. 2020. № 3 (13), P. 25–33.

11. Давидов В. В., Можаєв М. А., Ліцзян Джан. Аналіз та порівняльні дослідження методів підвищення рівня безпеки програмного забезпечення // Сучасні інформаційні технології. 2020. Том 4. № 3. С. 124–132.

12. Давидов В. В., Гребенюк Д. С. Метод первинного виділення хмарних обчислювальних ресурсів на основі аналізу ієрархій // Системи управління, навігації та зв'язку. 2020. Том 3 (61). С. 80-85.

13. Davydov V., Hrebeniuk D. Development the resources load variation forecasting method within cloud computing systems // Advanced Information Systems. 2020. Vol. 4. No. 4. P. 128–135.

14. Золотухіна О. А., Волошин Д. Г., Давидов В. В., Бречко В. О. Розробка імітаційної моделі процесу розрахунку і коригування безпечної польотної траєкторії безпілотного літального апарату // Телекомунікаційні та інформаційні технології. 2020. № 4 (69). С. 87–94.

15. Semenov S., Davydov V., Hrebeniuk D. Research of the software security model and requirements // Advanced Information Systems. 2021. Vol. 5. № 1. P. 87–92.

16. Semenov S., Liqiang Zhang, Weiling Cao, Davydov V. Development of protecting a software product methematical model from unlicensed copying based on the GERT method // Information Processing Systems. 2021. № 1 (164). P. 73-82.

17. Liqiang Zhang, Weiling Cao, Davydov V., Brechko V. Analysis and comparative research of the main approaches to the mathematical formalization of the penetration testing process // Information Processing Systems. 2021. № 2 (64). С.73-77.

18. Kuchuk N., Cherneva G., Davydov V. Method of packet fragmentation in unstable data exchange in computer networks on transport // Mechanics Transport Communications: Academic journal. Vol. 19. Is. 1. 2021. P. X1-1 – X1-7, Article № 2064.

19. Кучук Н. Г., Давидов В. В. Моделі і методи захисту інформаційних структур для комп'ютерних систем на інтегрованих програмних платформах (монографія). Харків, 2021. 160 с.

20. Давидов В. В. Моделі та методи підвищення безпеки програмного забезпечення (монографія). Харків, 2021. 146 с.

21. Semenov S., Davydov V., Semenova A., Voloshyn D., Lymarenko V. Method of UAVs Quasi-Autonomous Positioning in the External Cyber Attacks Conditions // 10th International Conference on Dependable Systems, Services and Technologies (DESSERT). 2019. (**SCOPUS**).

22. S. Semenov S., Davydov V., Voloshyn D. Obfuscated Code Quality Measurement // Metrology and metrology assurance 2019: 29th International Scientific Symposium. September 6-9, 2019. Sozopol, Bulgaria. P. 44-47. (**SCOPUS**).

23. Semenov S., Davydov V., Voloshyn D. Data Protection Method of an Unmanned Aerial Vehicle based on Obfuscation Procedure // CEUR Workshop Proceedings, 2020, Volume 2654. P. 515-525. (**SCOPUS**).

24. Давыдов В. В. Анализ методов обнаружения злоумышленного кода в Android приложениях // Інформаційні технології, наука, техніка, технологія, освіта, здоров'я : міжнар. наук.-практ. конф., тези доповідей. Харків, 2015. С. 36.

25. Davydov V., Zmiivska V., Shypova T., Lysytsia D. Analysis of fractal noise indicators in measuring systems of technical objects // Metrology and metrology

assurance 2018: 28th International Scientific Symposium, September 10-14, 2018 : Sozopol, Bulgaria. P. 44-47.

26. Kuchuk N., Shyman A., Hrebeniuk D., Davydov V. Mathematical model of the information system synthesis process // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: міжнародна наук.-техн. конф., матеріали. – Баку: ВА ЗС АР; Харків: НТУ «ХПІ»; Київ: НАУ; Харків: ДП «ПДПРОНДІАВІАПРОМ»; Жиліна: Університет, 2021. С. 21.

27. Семенов С. Г., Давыдов В. В., Мовчан А. В. Система формирования цифрового идентификатора программного обеспечения для защиты авторских прав // Сучасні проблеми інформатики в управлінні, економіці, освіті та подоланні наслідків Чорнобильської катастрофи: міжнарод. наук. сем., тези доповідей. Київ, 2016. С. 110-116.

28. Семенов С. Г., Давидов В. В. Оценка рисков контроля и диагностики технического состояния объектов критического применения // Metrology and Metrology assurance 2016. 26th National Scientific Symposium with international participation. Sozopol, Bulgaria. 2016. P. 395-399.

29. Semenov S., Hrebeniuk D., Davydov V. Software copyright protection using identification key // Aviation in the XXI-st Century: the 7th World Congress, September 19-21, 2016: Kyiv, Ukraine. P. 1.10.20-1.10.23.

30. Давыдов В. В. Процедура генерации лицензионного ключа для защиты авторских прав // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : міжнар. наук.-техн. конф., тези доповідей. Полтава, 2017. С. 52.

31. Давыдов В. В., Гребенюк Д. С. Особенности распределения ресурсов для многомашинных вычислительных комплексов // Проблеми інформатизації: міжнар. наук.-техн. конф., тези доповідей. Полтава, 2017. С. 21.

32. Кучук Н. Г., Давидов В. В., Гребенюк Д. С. Аналіз методів розрахунку розмірності мінковського для фрактального трафіка мультисервісної мережі e-learning // Проблеми інформатизації: міжнародна наук.-техн. конф., тези доповідей. Черкаси, 2018. С. 34.

33. Давидов В. В., Волошин Д. Г. Семенов С. Г. Про завдання позиціонування безпілотних літальних апаратів в умовах кібератак // Актуальні питання протидії кіберзлочинності та торгівлі людьми: Всеукраїнська наук.-практич. конф., збірник матеріалів. Харків, 2018. С. 311.

34. Давидов В. В., Гребенюк Д. С. Метод прогнозування навантаження ресурсів хмарних обчислюваних систем // Проблеми інформатизації: міжнародна наук.-технічн. конф., тези доповідей. Черкаси. 2020. С. 73.

35. S Semenov S., Davydov V., Weilin C., Liqiang Z., Petrovskaya I. Enhanced Software vulnerability model // Інформаційні технології і безпека : міжнародна наук.-практ. конф., матеріали. Київ, 2020. С. 56-60.

36. Semenov S., Bartosh M., Davydov V., Turuta O. Improvement of the Task Scheduler Model Taking Into Account the Heterogeneity of the Entities // Fifth International Scientific and Technical Conference "Computer and Information Systems and Technologies". 2021. P. 42-43.

АНОТАЦІЯ

Давидов В.В. Моделі та методи підвищення безпеки байт-код орієнтованого програмного забезпечення в умовах кібератак. – На правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю 05.13.05 – «Комп'ютерні системи та компоненти». – Черкаський державний технологічний університет, Черкаси, 2021.

Дисертаційна робота присвячена вирішенню актуальної науково-технічної проблеми підвищення безпеки програмного забезпечення в умовах кібератак у комп'ютерних системах на основі розробки моделей та методів обфускації програмного коду та створення ліцензійних ключів для захисту авторських прав. Для вирішення поставлених завдань у роботі: розроблено метод перевірки логіко-смислової подібності стандартних послідовних схем програм; розроблено GERT-модель процесу обфускації програмних модулів, що реалізує парадигми використання математичного апарату гамма-розподілу у якості ключового на всіх етапах моделювання процесу; удосконалено метод обфускації програмних модулів; розроблено критерій оцінки якості обфускації ПЗ; розроблено модель системи безпеки програмного забезпечення на основі математичного апарату моделювання GERT-мереж з використанням операцій безпечного переходу та кодування ліцензійних ідентифікаторів; вдосконалено метод формування цифрового ідентифікатора програмного забезпечення для захисту її авторських прав.

Результати дисертаційної роботи впроваджено в діяльність комерційних підприємств та навчальних закладах України.

Ключові слова: кібератаки, захист авторських прав, водяні знаки, обфускація, GERT-мережі, безпека програмного забезпечення, метрики якості коду, цифрові ідентифікатори, логіко-смислова подібність, Standalone-застосунки.

АННОТАЦИЯ

Давыдов В.В. Модели и методы повышения байт-код ориентированного программного обеспечения в условиях кибератак. – На правах рукописи.

Диссертация на соискание ученой степени доктора технических наук по специальности 05.13.05 – «Компьютерные системы и компоненты». – Черкасский государственный технологический университет, Черкассы, 2021.

Диссертация посвящена решению актуальной научно-технической проблемы повышения безопасности программного обеспечения в условиях кибератак в компьютерных системах на основе разработки моделей и методов обфускации кода, а также создания лицензионных ключей для защиты авторских прав. Для решения поставленных задач в работе: разработан метод проверки логико-смысловой подобности стандартных последовательных схем программ; разработана GERT-модель процесса обфускации программных модулей, которая реализует парадигмы использования математического аппарата гамма-распределения в качестве ключевого на всех этапах моделирования процесса; усовершенствован метод обфускации программных модулей; разработан критерий оценки качества обфускации программного обеспечения; разработана модель системы безопасности

программного обеспечения на основе математического аппарата моделирования GERT-сетей с использованием операций безопасного перехода и кодирования лицензионных идентификаторов; усовершенствован метод формирования цифрового идентификатора программного обеспечения для защиты ее авторских прав.

Результаты диссертационной работы внедрены в деятельность коммерческих предприятий и учебных заведениях Украины.

Ключевые слова: кибератаки, защита авторских прав, водяные знаки, обфускация, GERT-сети, безопасность программного, метрики качества кода, цифровые идентификаторы, логико-смысловая подобность, Standalone приложения.

ANNOTATION

Davydov V. Models and methods of increasing the bytecode-oriented software security of during cyberattacks. – On the rights of the manuscript.

The dissertation on competition of a scientific degree of the doctor of technical sciences on a specialty 05.13.05 – "Computer systems and components". – Cherkasy State Technological University, Cherkasy, 2021.

The dissertation is devoted to solving the current scientific and technical problem of improving the security of software during cyberattacks in computer systems based on the development the models and methods of software code obfuscation and the license keys creation for copyright protection.

A study and comparative analysis of bytecode-oriented software models and methods security was conducted, which showed that the existing models and methods of desktop software security at the system level of design do not fully eliminate the impact of threats to the corresponding security threats. Therefore, the existing modules and methods of software protection do not meet the quality requirements regulated by international standards and standards of Ukraine on software quality. It is shown that increasing the security of software affects other indicators of software quality, namely: portability, maintainability, performance.

A method for checking the logical and semantic similarity of standard sequential program schemes has been developed, which is based on finding the most similar in terms of thermal history paths in the program. This method uses a graph of agreed program paths as a reference structure. The iterative procedure that performs this calculation is the procedure for constructing the graph markup. The developed method allowed to pass to polynomial calculations of thermal history ways search at a stage of the graph marking for parallel calculations instead of existing exponential. This allowed to reduce the time of checking the code for logical and semantic similarity to determine the impact of the code obfuscation developed method on its correctness.

A GERT-model of the obfuscation process of software modules has been developed, which is based on the mathematical apparatus of the Gamma-distribution at all stages of modeling the obfuscation process. This allowed to achieve the model unification in terms of the GERT-network modification. The results of the study showed that for the developed mathematical model, when adding another obfuscation process, the variance of the execution time increases by 12%, and when it is removed from the system, it decreases to

13%. The mathematical expectation of execution time changes in a geometric progression – so, when removing 1 node there is a decrease in mathematical expectation by 9%, and when increasing by 1 node – an increase in mathematical expectation by 26%. This shows the insignificance of changes in the studied indicators in terms of model modification and confirms the hypothesis of model unification in terms of using the mathematical apparatus of the Gamma-distribution as the main one. This reduces the time to decide on the feasibility of using the obfuscation process in the use of flexible methodologies.

The method of safe transition in GERT-networks used as a graph of the control logic of the software product is developed. This logic is implemented depending on the identification or serial number. The methodology of scaling of the developed mathematical model by its horizontal and vertical scaling is developed. The expediency of using each type of scaling is shown taking into account the criticality of the execution time of the software security check based on the license identifiers. Thus, with vertical scaling, due to the use of a parallel data processing process, the data processing time has not changed. This gives the prerequisites for the use of this type of scaling at the critical time of the process, as well as the need to significantly increase the length of the license key on weak devices (e.g., embedded devices and Internet of Things). When scaling horizontally, due to the use of additional sequential data processing processes, the processing time increased significantly (3.43 times). However, the introduction of additional sequential actions increases the analysis time of the behavior algorithm. This gives the prerequisites for the use of this type of scaling when using application software, where the start time of the program is not critical.

A model of digital software identifier creation has been developed. A distinctive feature of created identifier is the individual data on end-user computer systems usage for unambiguous identification of accessories for which licensed software is installed in the process of licensing digital identifier creation. This makes it possible to increase the security of the software by protecting against unlicensed copying. An algorithm for the operation of the system and the generation of a license key, adapted to the input data and possible conditions for software verification, is proposed. Also, the model takes into account the possibility of embedding an arbitrary (specified by the developers) code in the body of the license key, which will be executed during verification. These manipulations have complicated the analysis and hacking of the licensed component of the software. This allowed to increase the average breaking time by 1.29 times.

Appropriate algorithms for software modules obfuscation and license keys generation have been developed and a software security system based on the subsystem of work with license identifiers and software code obfuscation has been implemented. A method for obtaining obfuscation metrics for a multi-project solution based on existing and improved code quality metrics has been developed, which allows to quantify the degree of code obfuscation. The study of the developed system allowed to increase the security of bytecode-oriented software by 1.4 times. The reliability and effectiveness of the proposed methods and models for improving the security of bytecode-oriented software.

Keywords: cyberattacks, copyright protection, watermarks, obfuscation, GERT-networks, software security, code quality metrics, digital identifiers, logical and semantic similarity, Standalone applications.