УДК 004.056.5 **DOI:10.30837/rt.2020.4.203.10**

К.Ю. ШЕХАНІН, Ю.І. ГОРБЕНКО, канд. техн. наук,
Л.О. ГОРБАЧОВА, О.О. КУЗНЕЦОВ, д-р техн. наук

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ НОСІЇВ ІНФОРМАЦІЇ ДЛЯ СТЕГАНОГРАФІЧНОГО ПРИХОВУВАННЯ ДАНИХ В КЛАСТЕРНИХ ФАЙЛОВИХ СИСТЕМАХ

Вступ

Із розвитком інформаційних технологій важливим стало питання як зберігати, передавати та оброблювати інформацію таким чином, щоб оптимізувати цей процес, тобто зменшивши витрати та підвищивши швидкодію та якість відповідних сервісів. Інколи інформація повинна бути захищена – цим займається галузь науки криптографія, яка гарантує доступ до інформації лише авторизованим користувачам. Але у разі використання криптографічних методів для неавторизованих користувачів (потенційних зловмисників) відомо про сам факт збереження або передачі інформації [1 – 3]. Це у подальшому може призвести до розкриття змісту інформації шляхом використання методів криптоаналізу. Щоб уберегти інформацію від розкриття використовують стеганографічні методи [4 – 6].

Стеганографія – це наука, яка забезпечує приховання самого факту обробки, зберігання чи передачі інформації, на відмінну від криптографії яка забезпечує лише захищеність змісту інформації [5]. Використання стеганографії доповнює криптографічні методи, тож ці методи є компліментарними один до одного з точки зору захищеності інформації. Як приклад можна згадати використання шифру Цезаря – це буде криптографічний метод, та використання «невидимих» чорнил – це буде стеганографічний метод [4, 5, 7]. А якщо сумісно використати ці два методи, то зловмиснику необхідно спочатку буде знайти текст, написаний невидимими чорнилами, а лише потім розшифрувати шифр Цезаря. Звісно, у сучасності данні методи не є ефективними. Наразі більшої популярності набувають методи комп'ютерної, або цифрової стеганографії [4, 5, 7 – 10]. Інформаційні повідомлення подаються у вигляді цифрових даних, які приховуються в інших цифрових даних (т.з. cover data). Cover files передаються та зберігаються у відкритому вигляді, це можуть бути, наприклад, цифрові зображення, які у великій кількості передаються сучасними каналами електронної пошти. Але вповноважений отримувач, який має секретний ключ, приймає cover file та може відновити таємне повідомлення. Звісно, що при цьому приховується сам факт існування таємного повідомлення, відслідкувати та дослідити всі cover files в інтернеті фізично неможливо [9 – 11].

Стеганографічні методи постійно вдосконалюються та розвиваються. Основна вимога до cover files є їхня надмірність (збитковість). Наприклад, надмірність цифрових зображень дозволяє приховати досить великі за обсягом інформаційні повідомлення.

Останніми роками набули розвитку методи технічної стеганографії. В таких системах приховування інформації досягається шляхом використання властивостей, які штучно зроблено людиною при побудові різних технічних засобів. Як приклад, можна навести мережеву стеганографію [12 – 16], в якій застосовуються різні особливості побудови сучасних телекомунікаційних систем та мереж, в тому числі штучна надмірність при визначенні форматів пакетів даних та способів їхньої передачі [17 – 19]. В 3D стеганографії використовуються надмірності цифрових 3D моделей та створених за їх допомогою фізичних об'єктів [20 – 24]. Наприклад, у роботах [25, 26] запропоновано приховувати інформацію шляхом створення фізичних об'єктів всередині інших (cover) об'єктів.

Ще одним прикладом технічної стеганографії є застосування особливостей побудови кластерних файлових систем. Зокрема, у роботах [27 – 29] запропоновано методи, які дозволяють ефективно приховувати інформацію шляхом зміни чергування (рос. – чередование) окремих кластерів т.з. покрівельних файлів. Імена (назви) таких файлів є ключовою інформацією і відновити приховане повідомлення без посилань (тобто без назв) покрівельних фай-

лів вкрай важко. Подальші дослідження кластерних файлових систем [30 – 36] дозволяють стверджувати, що застосування особливостей організації зберігання інформації дає змогу отримати надійний та безпечний механізм стеганографічного приховування інформаційних повідомлень. Звісно, що стійкість та швидкодія кластерних стеганосистем безпосередньо спирається на конкретні властивості файлової системи. Зокрема, безпека спирається на фрагментарність та переплетеність (в термінах робіт [27, 27, 30, 33]) покрівельних файлів, а швидкодія залежить від фрагментарності файлової системи та характеристик конкретного носія інформації. Отже, актуальним є питання аналізу різних носіїв інформації та відповідних файлових систем для можливого застосування в стеганографічних методах приховування, дослідження впливу окремих показників на ефективність кластерних стеганосистем.

Мета роботи – аналіз відомих технологій зберігання інформації, дослідження властивостей фрагментарності файлової системи та переплетеності окремих файлів на швидкість запису/зчитуванні інформації. Практичне значення отриманих результатів полягає у їх безпосередньому застосуванні до обґрунтування рекомендацій з побудови кластерних стеганосистем.

1. Методи приховування інформації у кластерні файлові системи

Файлова система встановлює порядок, спосіб організації, зберігання та іменування даних на носіях інформації в інформаційних системах, а також в іншому електронному обладнанні: цифрових фотоапаратах, мобільних телефонах і т.п. [37 – 39]. Файлова система визначає формат вмісту і спосіб фізичного зберігання інформації, яка групується у файли. Конкретна файлова система визначає розмір імен файлів і (каталогів), максимальний можливий розмір файлу і розділу, набір атрибутів файлу, тобто визначає метадані файлів. Деякі файлові системи надають сервісні можливості, наприклад, розмежування доступу або шифрування файлів.

Найпростіші методи приховування інформації у структурі файлових систем розглянуто в [32, 34 – 36]. Дані методи застосовують вільні кластери (або службові поля даних) для запису прихованої інформації, але такий спосіб, з точки зору конфіденційності інформації, є ненадійним [27, 29]. Інші методи, наприклад [27 – 31], ґрунтуються на використанні покриваючих файлів (cover file) і приховуванні інформаційних даних за допомогою взаємного перемішування файлів, тобто зміни відносних позицій кластерів декількох покриваючих файлів один щодо одного.

1.1. Приховування інформації через перемішування кластерів

Приховування інформації через перемішування кластерів різних cover files досліджено у роботах [27, 29]. Прихована інформація представляється у вигляді бітового масиву: $M = \{b_0, b_1, ..., b_{n-1}\}$, $b_i \in \{0,1\}$. На інформаційному носії обирають $p = 2^m$, $m \in N$ покриваючих файлів (cover files): $F_0, F_1, ..., F_{p-1}$. Порядок кластерів покриваючих файлів приховує інформаційне повідомлення, тобто після вбудовування покриваючи файли не можна модифікувати, видаляти та переміщувати. Натуральне число m та імена покриваючих файлів є секретним ключем. Важливий також порядок упорядкування cover files [29].

Формується масив номерів кластерів покриваючих файлів:

$$C = \begin{pmatrix} c_{0,0} & c_{0,1} & \dots & c_{0,L_0-1} & & & \\ c_{1,0} & c_{1,1} & \dots & \dots & \dots & \dots & c_{1,L_1-1} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ c_{p-1,0} & c_{p-1,1} & \dots & \dots & c_{p-1,L_{p-1}-1} & & \end{pmatrix},$$

де кожен рядок масиву містить номера кластерів відповідного файлу. Наприклад, файлу F_i відповідає i-й рядок масиву C, тобто номера кластерів i-го покриваючого файлу можуть бути подані у вигляді масиву $C_{F_i} = \{c_{i,0}, c_{i,1}, ..., c_{i,L_i-1}\}$, де L_i _ число кластерів в i-у покриваю-

чому файлі. Якщо при приховуванні інформації необхідно зберегти без зміни вміст покриваючих файлів, тоді потрібно, щоб виконувалися умови: $\forall i: L_i \geq k$, $k = n/m$.

Формується масив D номерів вільних кластерів файлової системи: $D = \{c_1, c_2, ..., c_{L_D}\}$, так щоб виконувалася умова $c_1 < c_2 < ... < c_{L_D}$. Число L_D дорівнює кількості вільних кластерів файлової системи, причому потрібно, щоб виконувалася умова: $L_D \geq \sum_{i=0}^{p-1} L_i$.

Інформаційне повідомлення M розбивається на блоки по m бітів: $M = \{B_1, B_2, ..., B_k\}$, де $k = \lceil n/m \rceil$ та якщо $k = n/m$, то $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …, $B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{km-1}\}$. Якщо $k < n/m$, то останній блок необхідно дописати неінформаційними значеннями, наприклад нулями, $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …, $B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{n-1}, \underbrace{0, 0, ..., 0}_{km-n}\}$. Кожен блок B_i, $i = 1, 2, ..., k$ подається як натуральне число, тобто $\forall i: 0 \leq B_i \leq p-1$. Кожне натуральне число B_i, $i = 1, 2, ..., k$ подається як номер покриваючого файлу з множини файлів $F_0, F_1, ..., F_{p-1}$.

Всі кластери покриваючих файлів перезаписуються у вільні кластери файлової системи, тобто масив D заповнюється номерами кластерів із масиву C. Порядок перезапису кластерів покриваючих файлів відповідає послідовності натуральних чисел $\{B_1, B_2, ..., B_k\}$, що задаються приховуваним повідомленням. Для прикладу, у перший вільний кластер записуємо перший кластер покриваючого файлу із номером B_1, у другий вільний кластер – наступний кластер покриваючого файлу із номером B_2 і так далі. Натуральні числа B_i можуть співпадати і в цьому разі записуємо наступний кластер того ж самого покриваючого файлу із номером B_i. Для посилення захисту від детектування та декодування стеганосистеми додатково можуть застосовуватися певні механізми, наприклад [29], обирається ключ ініціалізації B_0, а порядок перезапису кластерів покриваючих файлів задається послідовністю натуральних чисел $\{N_1, N_2, ..., N_k\}$, $N_i = B_{i-1} + B_i \bmod p$, $0 \leq N_i \leq p-1$. Тоді, у перший вільний кластер перезаписуються перший кластер покриваючого файлу із номером N_1, у другий вільним кластер – черговий кластер покриваючого файлу із номером N_2 і так далі.

В результаті виконання алгоритму перші k вільних кластерів файлової системи будуть записані кластерами покриваючих файлів. Отже, повинна виконуватися умова $k \leq L_D$.

Для вилучення прихованого повідомлення M формується масив D номерів кластерів покриваючих файлів: $D = \{c_1, c_2, ..., c_{L_D}\}$, причому необхідно, щоб виконувалась умова $c_1 < c_2 < ... < c_{L_D}$. Кожен номер кластеру з цього масиву співвідноситься тільки з одним кластером одного покриваючого файлу. Саме така відповідність пов’язана із логікою вбудовування інформації і використовується для вилучення прихованої інформації. Формується послідовність натуральних чисел $\{B_1, B_2, ..., B_k\}$, які відповідають блокам прихованого повідомлення: $B_1 = \{b_0, b_1, ..., b_{m-1}\}$, $B_2 = \{b_m, b_{m+1}, ..., b_{2m-1}\}$, …, $B_k = \{b_{(k-1)m}, b_{(k-1)(m+1)}, ..., b_{km-1}\}$. З цих бітових блоків формується інформаційне повідомлення $M = \{b_0, b_1, ..., b_{n-1}\}$, $b_i \in \{0,1\}$. Якщо $k < n/m$, тоді останній блок «обрізається» – його останні $km - n$ біт не несуть інформаційного значення.

Недоліком даного методу є незначний обсяг розміру прихованої інформації, який залежить від кількості покриваючих файлів та розміру покриваючих файлів у кластерах. У кожному кластері покриваючих файлів може бути приховано $\log_2 p = m$ інформаційних бітів.

1.2.Приховування інформації через додаткове перемішування кластерів кожного покрівельного файлу

Подальший розвиток розглянутого методу наведено у роботах [30, 31]. Для збільшення обсягу прихованого повідомлення в модифікованому методі запропоновано додаткове перемішування кластерів кожного cover files.

.

Модифікований метод приховування інформації в кластерних файлових системах ґрунтується на використанні одного або декількох покриваючих файлів (cover files) та приховуванні інформаційного повідомлення за шляхом зміни порядку позицій кластерів різних покриваючих файлів та зміні чергування кластерів у межах одного покриваючого файлу. На відміну від описаного методу, даний дозволяє досягти збільшення прихованої інформації на один біт за кожен кластер, при одних і тих значеннях ключових параметрів.

Приховувана інформація подається у вигляді бітового масиву: $M=\{b_0^*,b_1^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*,b_0,b_1,...,b_{n-1}\}$, $b_i^*,b_i\in\{0,1\}$. На інформаційному носії обирають $p=2^m$, $m\in N$ покриваючих файлів (cover files): $F_0,F_1,...,F_{p-1}$. Формується масив номерів кластерів покриваючих файлів як в описаному вище методі. Для кожного покриваючого файлу змінюється порядок чергування кластерів у кожному покриваючому файлі. Порядок чергування задається інформаційною послідовністю M. Для цього, формується p бітових масивів інформаційних бітів:

$$M_1=\{b_0^*,b_1^*,...,b_{L_1-1}^*\},$$
$$M_2=\{b_{L_1}^*,b_{L_1+1}^*,...,b_{L_1+L_2-1}^*\},$$
$$\dots,$$
$$M_{L_p}=\{b_{L_1+L_2+...+L_{p-1}}^*,b_{L_1+L_2+...+L_{p-1}+1}^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*\},$$

кожен з яких зіставляється із масивом номерів кластерів покриваючих файлів

$$C_{F_1}=\{c_{1,0},c_{1,1},...,c_{1,L_1-1}\},$$
$$C_{F_2}=\{c_{2,0},c_{2,1},...,c_{2,L_2-1}\},$$
$$\dots,$$
$$C_{F_p}=\{c_{p,0},c_{p,1},...,c_{p,L_p-1}\}.$$

Позиції кластерів кожного покриваючого файлу перезаписуються, тобто номери кластерів у кожному з масивів C_{F_1}, C_{F_2}, … , C_{F_p} змінюють своє чергування відповідно до значень бітових масивів M_1, M_2, … , M_{L_p}.

У результаті отримують нові масиви номерів кластерів $C_{F_1}^*$, $C_{F_2}^*$, … , $C_{F_p}^*$.

Перезапис позицій кластерів у кожному покриваючому файлі може здійснюватися різними способами. Наприклад, шляхом розбиття всіх номерів позицій кластерів $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$ на дві половини і співставлення кожної половини із значенням інформаційного біту. Тоді, наприклад, якщо $b_j^*=1$, $L_1+L_2+...+L_{i-1}-1<j\le L_1+L_2+...+L_i-1$, на j-у позицію в масиві $C_{F_i}^*$ розміщують кластер з першої половини впорядкованих номерів, якщо $b_0^*=0$ – з другої половини.

Сформовані таким чином масиви $C_{F_i}^*=\{c_{i,0}^*,c_{i,1}^*,...,c_{i,L_i-1}^*\}$ перезаписаних позицій номерів покриваючих файлів утворюють масив

$$C^*=\begin{pmatrix} c_{0,0}^* & c_{0,1}^* & ... & c_{0,L_0-1}^* & & & \\ c_{1,0}^* & c_{1,1}^* & ... & ... & ... & ... & c_{1,L_1-1}^* \\ ... & ... & ... & ... & ... & ... & ... \\ c_{p-1,0}^* & c_{p-1,1}^* & ... & ... & c_{p-1,L_{p-1}-1}^* & & \end{pmatrix}.$$

Зміна порядку чергування кластерів в кожному покриваючому файлі дозволяє приховати перші $L_1+L_2+...+L_{p-1}$ інформаційних бітів з масиву M, тобто інформаційну послідовність $\{b_0^*,b_1^*,...,b_{L_1+L_2+...+L_{p-1}-1}^*\}$. Решту n інформаційних бітів необхідно приховати так само, як і у розглянутому методі [29].

Формується масив D номерів вільних кластерів файлової системи: $D=\{c_1,c_2,...,c_{L_D}\}$, $c_1<c_2<...<c_{L_D}$. Послідовність із інформаційних бітів $\{b_0,b_1,...,b_{n-1}\}$ розбивається на блоки по m бітів кожен: $\{B_1,B_2,...,B_k\}$, кожен блок B_i, $i=1,2,...,k$ подається як натуральне число, тобто $\forall i: 0\le B_i\le p-1$. Кожне натуральне число B_i, $i=1,2,...,k$ подається як номер покриваючого файлу з множини файлів $F_0,F_1,...,F_{p-1}$.

Номери позицій кластерів покриваючих файлів перезаписуються у вільні кластери, тобто масив D заповнюється номерами кластерів з масиву $C*$ (перезаписаними кластерами, тобто із зміненим чергуванням кластерів у кожному покриваючому файлі). Порядок перезапису кластерів покриваючих файлів відповідає послідовності натуральних чисел $\{B_1,B_2,...,B_k\}$, які задаються приховуваним повідомленням. Наприклад, у перший порожній кластер перезаписуються перший кластер покриваючого файлу із номером B_1, у другий порожній кластер – черговий кластер покриваючого файлу із номером B_2 і т.д. Натуральні числа B_i можуть співпадати і в цьому разі записуються чергові кластери того ж самого покриваючого файлу із номером B_i. В результаті перші k вільних кластерів файлової системи будуть записані кластерами покриваючих файлів.

Для вилучення прихованого повідомлення M формується масив D номерів позицій кластерів покриваючих файлів: $D=\{c_1,c_2,...,c_{L_D}\}$. Кожен номер кластеру з цього масиву співвідноситися тільки з одним кластером одного покриваючого файлу. При цьому формується послідовність натуральних чисел $\{B_1,B_2,...,B_k\}$, які відповідають блокам прихованого повідомлення, тобто формується інформаційна послідовність $\{b_0,b_1,...,b_{n-1}\}$, $b_i\in\{0,1\}$.

Далі вилучається інформаційна послідовність $\{b_0^*,b_1^*,...,b^*_{L_1+L_2+...+L_{p-1}-1}\}$, $b_i^*\in\{0,1\}$.

Для цього аналізуємо масиви $C^*_{F_i}=\{c^*_{i,0},c^*_{i,1},...,c^*_{i,L_i-1}\}$ номерів кластерів кожного покриваючого файлу. Правило вилучення інформації відповідає логіці приховування. Наприклад, може застосовуватися розбиття всіх впорядкованих позицій номерів $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$ на дві половини і співставлення кожної половини із значенням інформаційного біту. Тоді, наприклад, якщо на j-й позиції в масиві $C^*_{F_i}$ розміщено кластер з першої половини масиву впорядкованих номерів $\{c_{i,0},c_{i,1},...,c_{i,L_i-1}\}$, приймають бітове значення $b^*_j=1$. Якщо з другої половини – приймають бітове значення $b^*_0=0$.

Таким чином, за рахунок додаткової зміни порядку чергування номерів кластерів у кожному із покриваючих файлів вдається підвищити обсяг прихованої інформації. Зокрема, в порівнянні із способом-прототипом, додатково вдається приховати по одному інформаційному біту на кожен кластер покриваючого файлу.

2. Аналіз відомих технологій зберігання інформації

У даній роботі проаналізовано носії інформації які використовують сучасні технології способу збереження інформації, а саме:

1. SSD (*Solid-State Drive*) – це енергонезалежний немеханічний запам'ятовуючий пристрій на основі мікросхем пам'яті та керуючому контролері;

2. HDD (*Hard (magnetic) Disk Drive*) – це енергонезалежний запам'ятовуючий пристрій оснований на принципі магнітного запису на диск;

3. Flash-USB (*Universal Serial Bus*) – це енергонезалежний запам'ятовуючий пристрій на основі мікросхем, має подібну структуру до SSD накопичувачів, але має менші показники вмістимості та швидкодії, тому здебільшого використовуються для перенесення інформації з пристрою на пристрій, та відокремленого збереження незначної кількості важливої інформації (паролі, ключи доступу, документи).

Так як Flash-USB є технологією подібною до SSD, то переваги й недоліки у даних технології будуть схожі. Але, Flash-USB обмежені максимальним об'ємом зберігаємої інформації, а також інтерфейсом USB (1.5 Мбіт/с – 5 Гбіт/с), що є значно повільнішим за SATA (1.5-6 Гбіт/с) чи M.2 PCIe (8-32 Гбіт/с), через які зазвичай підключають SSD накопичувачі до комп'ютерів. (https://3dnews.ru/623760).

Порівнюючи переваги та недоліки SSD та HDD накопичувачів, можна стверджувати, що SSD мають більше переваг ніж недоліків, зведена інформація наведена у табл. 1. Дані таблиці мають неточний характер через те, що кожен виробник має свої характеристики стосовно пристроїв SSD та HDD. Та одна й та ж технологія навіть у одного виробника може мати різні показники у залежності від цінової категорії пристрою. Головною метою було показати загальну тенденцію та різницю між SSD та HDD накопичувачами.

Таблиця 1

Порівняння основних характеристик SSD та HDD пристроїв

Архітектура накопичувача	SSD	HDD
Рівень шуму	майже відсутній	Значний, так як є рухомі частини
Механічна стійкість	висока	низька, при падінні рухомі частини можуть бути пошкодженні
Енергоспоживання	2-3 Ватт/годину	5-6 Ватт/годину
Магнітна чутливість	майже відсутня	Значна, так як електромагнітне поле безпосередньо впливає на магнітний диск
Розміри	менші розміри та вага	більші розміри через присутність рухомих частин
Паралельні операції	присутні, що значно пришвидшує запис/зчитування декількох файлів	відсутні
Швидкість запису/зчитування (Мб/с)	1000-3200/2000-4000	100-320/200-400
Ціна ($/Гб)	0.5	0.1
Циклів перезапису (разів)	10000	>100000
Вплив фрагментації на роботу	Майже відсутній, швидкість зчитування/запису не залежить від фрагментованості файлу	Значний, так як зчитування виконується однією оптичною головкою

Але з розвитком технологій можна зазначити, що SSD накопичувачі у порівнянні з аналогами п'ятирічної давнини мають значний приріст у показниках, у той час як показники HDD пристроїв майже не змінились.

Якщо ще у 2015 р. кількість HDD пристроїв значно переважала на ринку накопичувачів, то вже у 2020 р. кількість пристроїв HDD та SSD порівнялась. Зведена гістограма наведена на рис. 1.

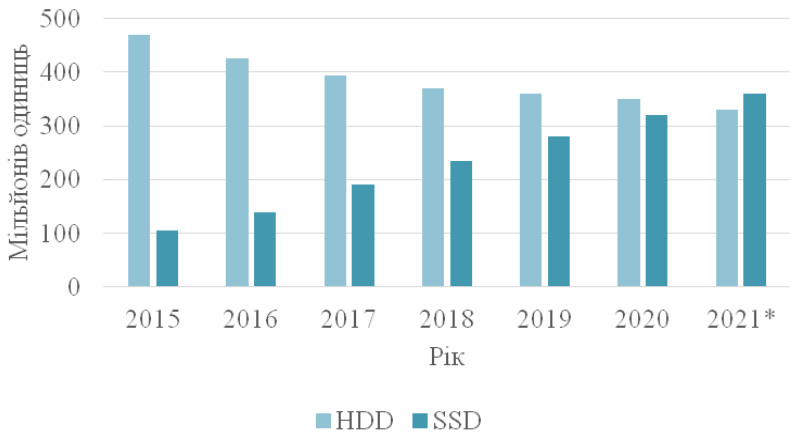

Рис. 1: Реалізація випущеної продукції накопичувачів інформації по всьому світі (у мільйонах одиниць)

Таким чином, можна стверджувати, що з кожним роком кількість SSD накопичувачів росте, забираючи долю ринка у HDD накопичувачів. Динаміка кількості HHD та SSD накопичувачів зазначена на рис. 2.

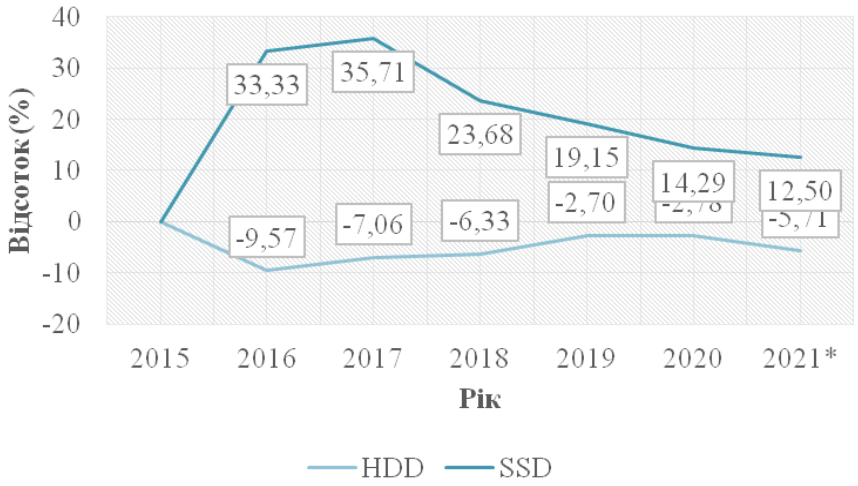

Рис. 2: Динаміка кількості HDD та SDD пристроїв

Для впливу технології накопичувача на можливості методу приховування інформації у структуру файлових систем [31, 33] більш детально оцінимо швидкодію кожної технології. Так як єдиної специфікації по SSD, HDD технологіям не існує, а швидкодія кожного пристрою залежить від виробника та цінової категорії, тож порівняльна оцінка зроблена по результатах електронного ресурсу UserBenchmark (https://www.userbenchmark.com). Даний ресурс збирає дані по компонентах комп'ютера (CPU, GPU, SSD, HDD, RAM, USB), проводячи тестування «opensource» програмним забезпеченням. Дані з UserBenchmark є об'єктивними, так як на ресурсі вказані результати більш ніж 160 мільйонів користувачів.

3. Методика дослідження та результати експериментів

У даній роботі нас цікавлять лише SSD та HDD накопичувачі. Так як для методу приховування інформації [31] важливим є параметр швидкості запису/зчитування з носія інформації, для цього виберемо найкращі пристрої за цими показниками. Для SSD накопичувачів це – Gigabyte GP-ASM2NE6100TTTD Aorus NVMe PCIe M.2 1TB (190$). Для HDD накопичувачів це – WD WD6001FZWX Black 6TB (225$).

Тестування даних пристрої проводилось методами:

1. Послідовного запису (*Sequential*) – це модель доступу до диску, при якій великі блоки даних записуються у сусідні блоки на поверхні пристрою з глибиною черги, що дорівнює одиниці. Даний термін використовується здебільшого у контексті порівняльного аналізу, швидкість вимірюють у МБ/с. Даний тип доступу часто використовують при зчитуванні/записі великих за розміром файлів, таких як відео, музика та зображення. Як демонструє практика, близько 50 % звичайного доступу користувача до диску на ПК буде складатись з послідовних операцій зчитування/запису. Накопичувачі, що в цілому використовуються для великих мультимедійних файлів та/або резервних копій, повинні мати відносно велику швидкість послідовного доступу до файлів;

2. Випадковий запис 4K (*Random 4k*) – це шаблон доступу до диску, при якому невеликі (4k) блоки даних записуються у випадкові місця на поверхні тестованого пристрою з глибиною черги, що дорівнює одиниці. Даний термін використовується здебільшого у контексті порівняльного аналізу, швидкість вимірюють у МБ/с. Даний метод оцінки швидкодії можна використовувати з урахування того, наскільки ефективно пристрій зчитує/записує невеликі фрагменти даних із випадкових місць. Даний шаблон значно розповсюджений під час запуску операційної системи, коли з накопичувача необхідно зчитати велику кількість файлів

конфігурації та драйверів. Як демонструє практика, близько 20 % звичайного доступу користувача до диску на ПК буде складатись з запису/зчитування з випадкових блоків накопичувача.

Результат тестування *Random 4k* є більш важливим для методу приховування інформації шляхом перемішування кластерів у структурі файлової системи, так як при приховуванні даних запис виконується у «випадкові» блоки накопичувача у залежності від приховуваної інформації, що відповідає швидкості доступу до файлу із значним рівнем фрагментації.

Також необхідно зазначити, що модифікований метод приховування даних у структуру файлової системи FAT (який запропоновано у [31]]) дозволяє використовувати 1 покриваючий файл з неперервним ланцюгом кластерів але з певним рівнем переплетеності.

Переплетеність – це властивість файлу, при якій кластери файлу розміщуються неперервно без проміжків між кластерами, але можливий зворотній напрямок у індексації кластерів. Як приклад, переплетений файл – це файл, кластери якого розміщені з 1 по 10 кластери, але ланцюг кластерів може мати вигляд як $Chain_F = [1,9,2,3,10,4,8,7,5,6]$. У загальному вигляді формула рівня переплетеності файлу:

$$Ent_F = \sum_{i=1}^{F_{len}} \left| Chain_F[i] - Chain_F[i+1] \right| - 1, \quad (1)$$

де Ent_F (*entanglement*) – рівень переплетеності файлу F; F_{len} – довжина файлу F у кластерах; $Chain_F[i]$ – i елемент ланцюгу кластерів файлу F.

Таким чином, рівень переплетеності файлу у попередньому прикладі

$$Ent_F = (|1-9|-1) + (|9-2|-1) + \ldots + (|5-6|-1) = 28.$$

Оскільки у даній роботі проаналізовано фізичні властивості носіїв інформації, то можна прирівняти вплив фрагментованості файлу до впливу переплетеності файлу на швидкість запису/зчитування. Адже що фрагментованість, що переплетеність так само змушує далі зміщувати головку зчитуючого пристрою (для HDD технології) для доступу до наступного кластеру.

Зведені результати тестувань пристроїв наведені у табл. 2 та 3.

Таблиця 2

Зведені результати швидкодії SSD та HDD пристроїв
за результатами UserBenchmark методом Sequential

Т	SSD (МБ/с)			HDD (МБ/с)		
Мт	Sequential			Sequential		
	Min.	Avg.	Max.	Min.	Avg.	Max.
R	809	1950	2318	102	167	215
W	1705	3115	3783	138	202	246
M	704	1998	2315	67.5	102	215

Таблиця 3

Зведені результати швидкодії SSD та HDD пристроїв
за результатами UserBenchmark методом Random 4k

Т	SSD (МБ/с)			HDD (МБ/с)		
Мт	Random 4k			Random 4k		
	Min.	Avg.		Min.	Avg.	
R	33	49.5	R	33	49.5	R
W	103	176	W	103	176	W
M	50.7	78.2	M	50.7	78.2	M

Примітка: Т – технологія пристрою; Мт – метод оцінювання; R (read) – швидкість зчитування даних; W (write) – швидкість запису даних; M (mixed) – швидкість почергового зчитування/запису даних.

Таким чином, якщо порівнювати швидкості накопичувачів, то безпосередньо SSD є більш швидким, у абсолютних значеннях. Якщо порівнювати спад швидкості Random 4k та Sequential, то SSD при фрагментованому записі має швидкість у 5 – 10 % у той час як HDD втрачають свою швидкість до 1,5 – 3 %. Звісно 5 – 10 % це мало, але значно краще ніж у HDD. Наведемо вплив фрагментованості носія на швидкість доступу до файлу. Для цього проаналізуємо середній час запису та читання даних з диску з фрагментацією та без.

Середній час читання з диска це час у секундах, який у середньому необхідний на одну операцію читання даних з носія інформації, результати вказані на рис. 3 (https://club.directum.ru/post/140).

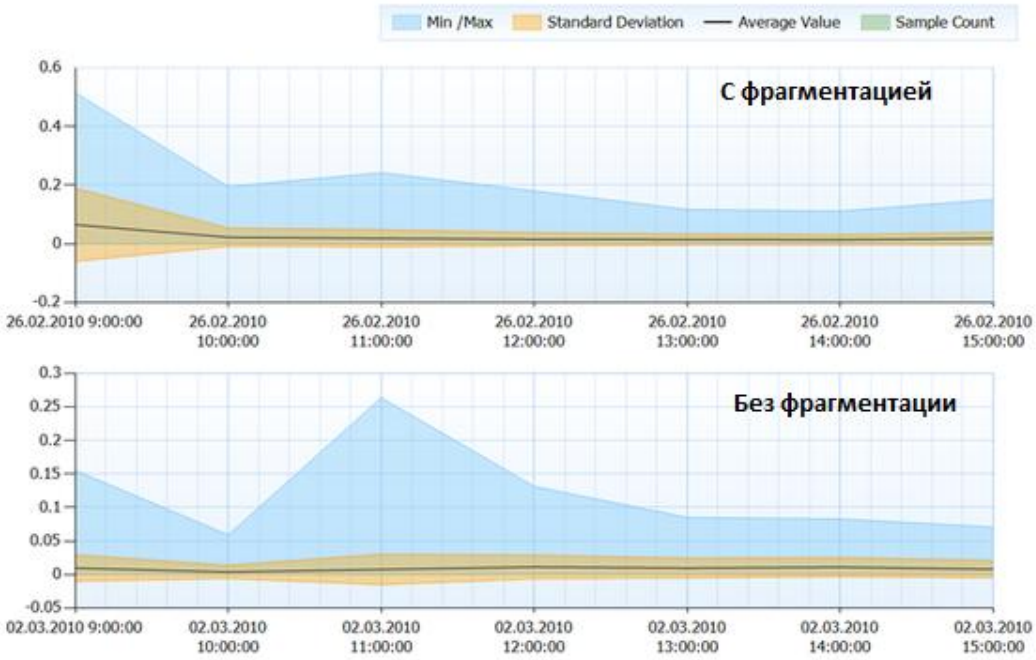

Рис. 3: Середній час зчитування даних з диску при фрагментації та без фрагментації

Результати наведені у табл. 4.

Таблиця 4

Оцінка затраченого часу при читанні даних з диску

Операція	Мінімальне значення (с)	Середнє значення (с)	Максимальне значення (с)
З фрагментацією	0	0.23	0.5
Без фрагментації	0	0.08	0.26

Також проаналізуємо середній час запису на диск. Так само, оцінюючи час при фрагментованості та без фрагментованості. Результати показані на рис. 4 та в табл. 5. (https://club.directum.ru/post/140).

Порівнюючи результати обробки даних з фрагментацією та без фрагментації можна стверджувати, що обробка даних без фрагментації ефективніша на 40 – 60 %, ніж обробка фрагментованих даних. Але використання SSD здатне зменшити негативний вплив фрагментованості даних на швидкість обробки цих даних (https://club.directum.ru/post/140).

Виходячи з результатів методів *Random 4k* та *Sequential*, можна стверджувати, що чим більший рівень фрагментації, тим менша швидкість доступу до файлу. Особливо це впливає на швидкодію HDD накопичувачів, у той час як SSD накопичувачі дозволяють проводити запис/зчитування файлиа майже без втрати швидкості, навіть при значному рівні фрагментованості.

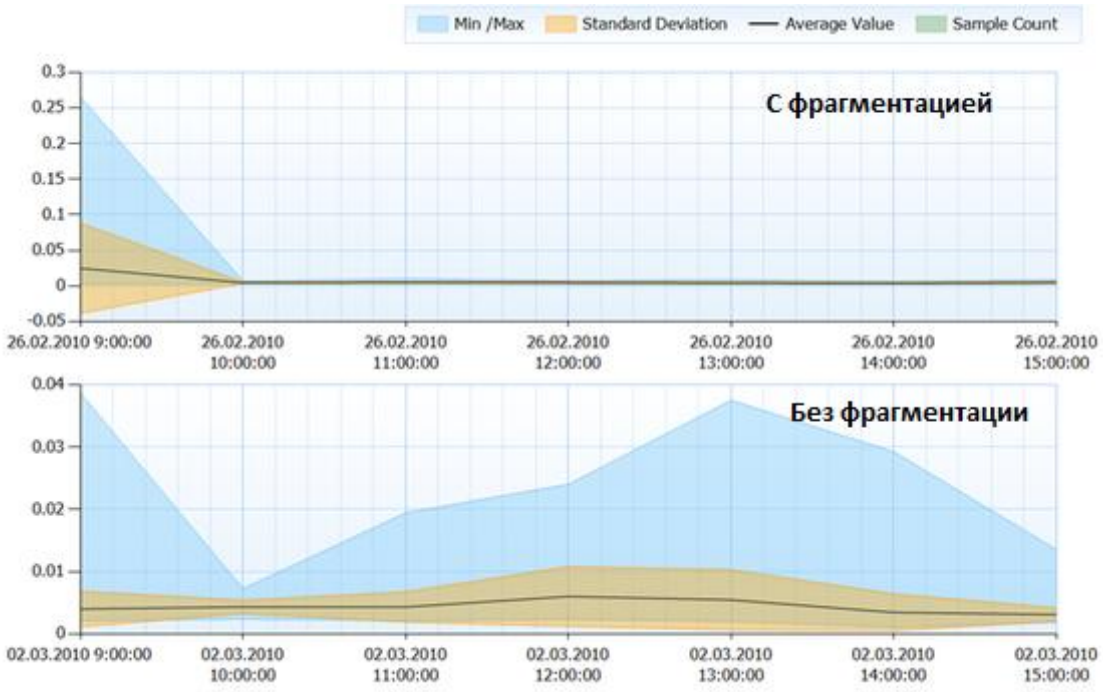

Рис. 4. Середній час запису даних на диску при фрагментації на без фрагментації

Таблиця 5

Оцінка затраченого часу при запису даних на диск

Операція	Мінімальне значення (с)	Середнє значення (с)
З фрагментацією	0,001787	0,007211
Без фрагментації	0,001044	0,004368

Висновки та рекомендації

Отже, можна стверджувати, що технологія SSD має значні переваги, завдяки чому накопичувачі з даною технологією стають більш розповсюдженими. SSD має у рази більшу швидкість доступу до файлів/секторів, навіть при значному показнику фрагментованості – це сприятливий показник для використання стеганографічних методів приховування даних. Також SSD має обмежену кількість циклів перезапису, і хоча це й недолік, але для методу приховування даних цей показник сприятливий, адже виконання дефрагментації є небажаною операцією для SSD.

Отже як висновок можна рекомендувати використання SSD носіїв інформації для приховування повідомлення у структуру файлової системи:

– через те що швидкість доступу до кластеру значно вища, що забезпечить більше швидке виконання стеганографічного методу;

– при збільшенні рівня фрагментованості (переплетеності) швидкість доступу до файлу втрачає не так багато, як у порівнянні з HDD технологією, що є значно важливішим показником при використанні методу приховування даних у структурі файлової системі;

– виконання дефрагментації SSD накопичувачів э небажаною процедурою, що призводить до збільшення загального рівня фрагментованості на носії інформації, що у свою чергу дозволяє приховати більше інформації без ризику розкриття (чим більший рівень фрагментованості на носії, тим більше інформації можна приховати [33]).

Таким чином, завдяки розповсюдженню SSD метод приховування інформації у структуру файлових систем шляхом перемішування кластерів покриваючих файлів є актуальним.

Список літератури:

1. Klima R.E., Klima R., Sigmon N.P., Sigmon N., Klima R., Sigmon N.P., Sigmon N. Cryptology: Classical and Modern. Chapman and Hall/CRC (2018). https://doi.org/10.1201/9781315170664.
2. Delfs H., Knebl H. Introduction to Cryptography. Springer Berlin Heidelberg. Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47974-2.
3. Childs L.N. Cryptology and Error Correction: An Algebraic Introduction and Real-World Applications. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-15453-0.
4. Manoj I.V.S. Cryptography and Steganography // IJCA. 1, 63–68 (2010). https://doi.org/10.5120/257-414.
5. Yahya A. Introduction to Steganography // Yahya A. (ed.) Steganography Techniques for Digital Images. pp. 1–7. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-78597-4_1.
6. Qin J., Luo Y., Xiang X., Tan Y., Huang H.: Coverless Image Steganography: A Survey // IEEE Access. 7, 171372–171394 (2019). https://doi.org/10.1109/ACCESS.2019.2955452.
7. Schöttle P., Böhme R. Game Theory and Adaptive Steganography. IEEE Transactions on Information Forensics and Security. 11, 760–773 (2016). https://doi.org/10.1109/TIFS.2015.2509941.
8. Yahya A. Steganography Techniques // Yahya, A. (ed.) Steganography Techniques for Digital Images. pp. 9–42. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-78597-4_2.
9. Fridrich J. Steganography in Digital Media: Principles, Algorithms, and Applications. Cambridge University Press, Cambridge; New York (2009).
10. Cox I., Miller M., Bloom J., Fridrich J., Kalker T. Digital Watermarking and Steganography. 2nd Ed. Morgan Kaufmann, Amsterdam, Boston (2007).
11. Kim C.R., Lee S.H., Lee J.H., Park J.-I. Blind decoding of image steganography using entropy model // Electronics Letters. 54, 626–628 (2018). https://doi.org/10.1049/el.2017.4276.
12. Rowland C.H. Covert channels in the TCP/IP protocol suite, https://firstmonday.org/ojs/index.php/fm/article/download/528/449?inline=1, last accessed 2020/11/08.
13. Mazurczyk W., Lubacz J. LACK – a VoIP steganographic method // Telecommun Syst. 45, 153–163 (2010). https://doi.org/10.1007/s11235-009-9245-y.
14. Lubacz J., Mazurczyk W., Szczypiorski K. Principles and Overview of Network Steganography // IEEE Communications Magazine. 52, (2012). https://doi.org/10.1109/MCOM.2014.6815916.
15. Mazurczyk W., Smolarczyk M., Szczypiorski K. On information hiding in retransmissions // Telecommun Syst. 52, 1113–1121 (2013). https://doi.org/10.1007/s11235-011-9617-y.
16. Cauich E., Gómez Cárdenas R., Watanabe R. Data Hiding in Identification and Offset IP Fields // Ramos, F.F., Larios Rosillo, V., and Unger, H. (eds.) Advanced Distributed Systems. pp. 118–125. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/11533962_11.
17. Wang M., Gu W., Ma C. A Multimode Network Steganography for Covert Wireless Communication Based on BitTorrent, https://www.hindawi.com/journals/scn/2020/8848315/, last accessed 2020/11/08. https://doi.org/10.1155/2020/8848315.
18. Seo J.O., Manoharan S., Mahanti A. Network steganography and steganalysis – a concise review // 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT). pp. 368–371 (2016). https://doi.org/10.1109/ICATCCT.2016.7912025.
19. Noskov A. Analysis of Network Protocols: The Ability of Concealing the Information // Computer and Network Security. (2020). https://doi.org/10.5772/intechopen.88098.
20. A High Capacity 3D Steganography Algorithm, https://www.computer.org/csdl/journal/tg/2009/02/ttg2009020274/13rRUwdIOUD, last accessed 2020/11/08. https://doi.org/10.1109/TVCG.2008.94.
21. Paramasivan T., Natarajan V., Gnanasekaran A., Venkatesan V., Anitha R. Pattern based 3D image Steganography. 3D Research. 4, (2014). https://doi.org/10.1007/3DRes.01(2013)1.
22. Chao M.-W., Lin C., Yu C.-W., Lee T.-Y. A high capacity 3D steganography algorithm // IEEE Trans Vis Comput Graph. 15, 274–284 (2009). https://doi.org/10.1109/TVCG.2008.94.
23. Li N., Hu J., Sun R., Wang S., Luo Z. A High-Capacity 3D Steganography Algorithm With Adjustable Distortion // IEEE Access. 5, 24457–24466 (2017). https://doi.org/10.1109/ACCESS.2017.2767072.
24. Thiyagarajan P., Natarajan V., Aghila G., Prasanna Venkatesan V., Anitha R. Pattern based 3D image Steganography. 3D Res. 4, 1 (2014). https://doi.org/10.1007/3DRes.01(2013)1.
25. Kuznetsov A., Stefanovych O., Gorbenko Y., Smirnov O., Krasnobaev V., Kuznetsova K. Information Hiding Using 3D-Printing Technology // 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). pp. 701–706 (2019). https://doi.org/10.1109/IDAACS.2019.8924352.
26. Kuznetsov A.A., Stefanovych O.O., Prokopovych-Tkachenko D.I., Kuznetsova K.O. 3D STEGANOGRAPHY INFORMATION HIDING. TRE. 78, (2019). https://doi.org/10.1615/TelecomRadEng.v78.i12.30.
27. Khan H., Javed M., Mirza F., Khayam S. Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel. (2012).

28. Khan H., Javed M., Khayam S.A., Mirza F. Designing a cluster-based covert channel to evade disk investigation and forensics. Computers & Security. 30, 35–49 (2011). https://doi.org/10.1016/j.cose.2010.10.005.
29. Venčkauskas A., Morkevicius N., Petraitis G., Ceponis J. Covert Channel for Cluster-based File Systems Using Multiple Cover Files // Information technology and control. 42, (2013). https://doi.org/10.5755/j01.itc.42.3.3328.
30. Kuznetsov A., Shekhanin K., Kolhatin A., Mikheev I., Belozertsev I. Hiding data in the structure of the FAT family file system // 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). pp. 337–342 (2018). https://doi.org/10.1109/DESSERT.2018.8409155.
31. Shekhanin K.Y., Kolhatin A.O., Demenko E.E., Kuznetsov A.A.: ON HIDING DATA INTO THE STRUCTURE OF THE FAT FAMILY FILE SYSTEM. TRE. 78, (2019). https://doi.org/10.1615/TelecomRadEng.v78.i11.50.
32. Vokorokos L., Madoš B., Ádám N., Baláž A., Porubän J., Chovancová E. Multi-Carrier Steganographic Algorithm Using File Fragmentation of FAT FS. COMPUTING AND INFORMATICS. 38, 343-366–366 (2019).
33. Shekhanin K., Kuznetsov A., Krasnobayev V., Smirnov O. Detecting Hidden Information // FAT. IJCNIS. 12, 33–43 (2020). https://doi.org/10.5815/ijcnis.2020.03.04.
34. Aycock J., de Castro D.M.N. Permutation Steganography in FAT Filesystems // Shi, Y.Q. (ed.) Transactions on Data Hiding and Multimedia Security X. pp. 92–105. Springer, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46739-8_6.
35. Davis J., MacLean J., Dampier D. Methods of Information Hiding and Detection in File Systems. (2010). https://doi.org/10.1109/SADFE.2010.17.
36. Neuner S., Voyiatzis A.G., Schmiedecker M., Brunthaler S., Katzenbeisser S., Weippl E.R. Time is on my side: Steganography in filesystem metadata // Digital Investigation. 18, S76–S86 (2016). https://doi.org/10.1016/j.diin.2016.04.010.
37. FAT File System, https://www.keil.com/pack/doc/mw/FileSystem/html/fat_fs.html, last accessed 2020/11/08.
38. FAT File Systems. FAT32, FAT16, FAT12 – NTFS.com, https://www.ntfs.com/fat_systems.htm, last accessed 2020/11/08.
39. Overview of FAT, HPFS, and NTFS File Systems, https://support.microsoft.com/en-us/help/100108/overview-of-fat-hpfs-and-ntfs-file-systems, last accessed 2020/11/08.

Надійшла до редколегії 05.11.2020

Відомості про авторів:

Шеханін Кирил Юрійович – аспірант кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук, Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: kyryl.shekhanin@nure.ua, ORCID: https://orcid.org/0000-0002-1441-7814

Горбенко Юрій Іванович – канд. техн. наук, перший заступник головного конструктора, АТ «Інститут інформаційних технологій», Україна; e-mail: gorbenkou@iit.kharkov.ua, ORCID: https://orcid.org/0000-0003-0073-9107

Горбачева Людмила Олегівна – студентка кафедри моделювання систем і технологій, факультет комп'ютерних наук, Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: lusyag23@gmail.com, ORCID: https://orcid.org/0000-0002-6053-7235

Кузнецов Олександр Олександрович – д-р техн. наук, професор, професор кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук, Харківський національний університет імені В.Н. Каразіна, Україна; e-mail: kuznetsov@karazin.ua, ORCID: https://orcid.org/0000-0003-2331-6326