**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА** **КОМП’ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ**

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Казмірчук

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

На правах рукопису

УДК 004.056.5:510.22(043.3)

**МАГІСТЕРСЬКА АТЕСТАЦІЙНА РОБОТА**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**

**«МАГІСТР»**

**Тема**:Модель безпеки комп’ютерної системи на базі технології блокчейн

|  |  |
| --- | --- |
| **Автор:** | Є.М. Шуневич |
| **Науковий керівник:** к.т.н., доц. | М.Б. Гумен |
| **Нормоконтролер:** асист. | С.В. Єгоров |

**Київ 2020**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет:** Кібербезпеки, комп’ютерної та програмної інженерії

**Кафедра:** Комп’ютеризованих систем захисту інформації

**Освітній ступінь:** Магістр

**Спеціальність:** 125 «Кібербезпека»

**Освітньо-професійна програма**: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Казмірчук

«\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**

**на виконання магістерської атестаційної роботи**

**магістранта Шуневича Євгенія Михайловича**

1. Тема: *Модель безпеки комп’ютерної системи на базі технології блокчейн*

затверджена наказом ректора від «02» жовтня 2019 р. № 2265/ст*.*

1. Термін виконання з 14.10.2019 р. по 09.02.2020 р*.*
2. Вихідні дані: проаналізувати існуючі моделі безпеки комп’ютерних систем; дослідити принципи роботи технології блокчейн та виявити її основні переваги та недоліки; розробити модель безпеки комп’ютерної системи на базі технології блокчейн.
3. Зміст пояснювальної записки: аналіз існуючих моделей безпеки комп’ютерних систем; технологія блокчейн, її принципи роботи, види, переваги та недоліки; розробка моделі комп’ютерної системи на базі технології блокчейн.

**КАЛЕНДАРНИЙ ПЛАН**

**виконання магістерської роботи**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Етапи виконання магістерської роботи** | **Термін виконання етапів** | **Примітка** |
|  | Уточнення постановки задачі | 15.10.19 – 20.10.19 | *Виконано* |
|  | Аналіз літературних джерел | 21.10.19 – 30.10.19 | *Виконано* |
|  | Обґрунтування вибору рішення | 30.10.19 – 05.11.19 | *Виконано* |
|  | Дослідження сучасних моделей безпеки комп’ютерних систем | 06.11.19 – 30.11.19 | *Виконано* |
|  | Аналіз принципів роботи технології блокчейн | 30.11.19 – 12.12.19 | *Виконано* |
|  | Розробка моделі комп’ютерної системи | 13.12.19 – 20.12.19 | *Виконано* |
|  | Розробка програмного забезпечення для реалізації моделі | 21.12.19 – 17.01.20 | *Виконано* |
|  | Перевірка на антиплагіат |  |  |
|  | Оформлення і друк пояснювальної записки |  |  |
|  | Оформлення презентації |  |  |
|  | Отримання рецензій від рецензента |  |  |
|  | Захист в ЕК |  |  |

Магістрант Є. Шуневич

(підпис, дата)

Науковий керівник М. Гумен

(підпис, дата)

**РЕФЕРАТ**

**Магістерська атестаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків і має 122 сторінки основного тексту, 42 рисунка, 25 таблиць, 13 сторінок додатків. Список використаних джерел містить 27 найменувань і займає 2 сторінки. Загальний обсяг роботи 135 сторінок.**

Метою роботи є підвищення рівня безпеки комп’ютерної системи за рахунок впровадження технології блокчейн в модель безпеки.

В роботі вирішено задачу побудови моделі безпеки комп’ютерної системи на базі технології блокчейн.

В роботі розроблено модель безпеки комп’ютерної системи, програмне забезпечення для реалізації суб’єктно-об’єктної моделі доступу та впровадження технології блокчейн та оцінено захищеність комп’ютерної системи.

Розроблена модель та програмне забезпечення відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності системи відслідковування ланцюга доставки медичних препаратів.

Можливі напрямки розвитку цієї роботи пов’язані із розширенням моделі і алгоритму програмного забезпечення відповідно до вимог міжнародних стандартів, наприклад ISO 27001, для більш повного аналізу та оцінки ризиків.

Ключові слова: блокчейн, модель безпеки комп’ютерної системи, інформаційна безпека, інформаційно-комунікаційна система, смарт-контракт, актив, загроза, оцінюючі компоненти.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6](#_Toc30359711)

[ВСТУП 7](#_Toc30359712)

[Розділ 1. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ БЕЗПЕКИ КОМП’ЮТЕРНИХ СИСТЕМ 10](#_Toc30359713)

[1.1. Основні положення в сфері захисту інформації 10](#_Toc30359714)

[1.2. Поняття моделей безпеки комп'ютерних систем 11](#_Toc30359715)

[1.3. Представлення інформації в автоматизованій системі 20](#_Toc30359716)

[1.4. Суб' єктно - об' єктне моделювання 21](#_Toc30359717)

[1.4.1. Формальні моделі управління доступом. 23](#_Toc30359718)

[1.4.2. Формальні моделі цілісності. 27](#_Toc30359719)

[1.5. Висновки по розділу 31](#_Toc30359720)

[Розділ 2: ТЕХНОЛОГІЯ БЛОКЧЕЙН, ЇЇ ПРИНЦИПИ РОБОТИ, ВИДИ, ПЕРЕВАГИ ТА НЕДОЛІКИ 34](#_Toc30359721)

[2.1 Складові технології блокчейн 34](#_Toc30359722)

[2.2 Основні принципи роботи технології блокчейн 39](#_Toc30359723)

[2.3. Класифікація блокчейнів. 43](#_Toc30359724)

[2.4. Переваги та недоліки технології блокчейн 50](#_Toc30359725)

[2.5. Висновки по розділу 53](#_Toc30359726)

[Розділ 3: РОЗРОБКА МОДЕЛІ КОМП’ЮТЕРНОЇ СИСТЕМИ НА БАЗІ ТЕХНОЛОГІЇ БЛОКЧЕЙН 54](#_Toc30359727)

[3.1 Аналіз моделі інформаційних потоків 54](#_Toc30359728)

[3.2 Розробка веб-платформи відслідковування ланцюга доставки медичних препаратів 56](#_Toc30359729)

[3.2.1 Імплементація блокчейн смарт-контракту. 57](#_Toc30359730)

[3.2.2 Імплементація веб-застосунку. 64](#_Toc30359731)

[3.3 Програмно-апаратні засоби захисту системи 80](#_Toc30359732)

[3.4 Оцінка захищеності комп’ютерної системи 81](#_Toc30359733)

[3.5 Висновки по розділу 82](#_Toc30359734)

[ВИСНОВКИ 84](#_Toc30359735)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 85](#_Toc30359736)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

КС – комп’ютерна система

АС – автоматизована система

МБЗ – монітор безпеки звернень

НСД – несанкціонований доступ

ІБ – інформаційна безпека

НД – нормативний документ

ТЗІ – технічний захист інформації

# ВСТУП

**Актуальність.** Зі стрімким розвитком інформаційних технологій зростає необхідність у розробці сучасних методів захисту інформації та систем, в якій вона зберігається. Поряд з цим зловмисники вдосконалюють свої засоби для атак на системи заради своєї вигоди. Дійсно, універсальних методів захисту не існує, багато в чому успіх при побудові механізмів безпеки для реальної системи буде залежати від її індивідуальних особливостей, облік яких погано подається формалізації. Тому часто інформаційну безпеку розглядають як деяку сукупність неформальних рекомендацій по побудові систем захисту інформації того або іншого типу.

Згідно з останніми дослідженням інституту Понемону та компанії IBM за 2019 рік, один інцидент з порушенням цілісності, конфіденційності або доступності даних коштував компаніям в середньому 3.92 мільйони доларів. В 2018 році ця цифра сягала 3.86 мільйонів, тому з кожним роком цінність даних невпинно збільшується [1].

Однією з найбільш перспективних та іноваційних підходів для зберігання даних на сьогодні є технологія “Блокчейн”, яка успішно зарекомендувала себе в таких сферах як криптовалюта, онлайн-голосування та публічні реєстри. Основною перевагою цієї технології є неможливість підробки даних після внесення їх в блокчейн.

Варто зазначити, що блокчейн-технології поступово впроваджуються і в Україні. Державне агентство з питань електронного урядування України підписали меморандум про співпрацю з компанією Bitfury щодо створення екосистеми для розвитку блокчейну в Україні [2].

В цій роботі проводиться класифікація блокчейнів за різними характеристиками, розглядаються поняття моделі безпеки та блокчейну, поширені моделі управління доступом, що дозволяють реалізувати формальний аналіз систем захисту, та пропонується практична реалізація моделі захисту комп’ютерної системи на базі технології блокчейн на прикладі платформи для відслідковування ланцюга доставки фармацевтичних препаратів.

**Відомі підходи до вирішення поставленої задачі.** На сьогоднішній день існує велика кількість блокчейн-рішень, що дозволяють зберігати цілісність даних: криптовалюти, які за допомогою блокчейну зберігають дані про транзакції між усіма користувачами мережі, публічні реєстри які містять у собі інформацію, що захищена блокчейном і доступна усім користувачам.

**Метою роботи** є підвищення рівня безпеки комп’ютерної системи за рахунок впровадження технології блокчейн в модель безпеки. Для досягненні мети вирішуються такі задачі:

* Дослідження відомих моделей безпеки
* Аналіз технології блокчейн, її видів, недолік та переваг
* Розробка та інтеграція програмного забезпечення на базі технології блокчейн що реалізує розроблену модель безпеки

**Галузь застосування.** Розроблена модель та програмне забезпечення відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності системи відслідковування ланцюга доставки медичних препаратів.

**Об’єктом дослідження** є процес захисту інформації в комп’ютерних системах.

**Предметом дослідження** є методи,моделі та алгоритми захисту інформації в комп’ютерних системах.

**Методи дослідження -** аналіз та класифікація технології блокчейн, об’єктно-орієнтоване програмування

**Науковою новизна одержаних результатів полягає в наступному:**

Удосконалено класичну модель безпеки шляхом інтеграції в неї технології блокчейн через практичну реалізацію смарт-контракту на мові програмування Solidity.

**Практичне значення отриманих результатів:**

Розроблено веб-застосунок для системи відслідковування ланцюга доставки медичних препаратів

**Апробація.** Основні положення роботи доповідалися і обговорювались на міжнародній інтернет-конференції “Інновації в науці: сучасні виклики” 07 жовтня 2019 р.

# Розділ 1. АНАЛІЗ ІСНУЮЧИХ МОДЕЛЕЙ БЕЗПЕКИ КОМП’ЮТЕРНИХ СИСТЕМ

## 1.1. Основні положення в сфері захисту інформації

Інформаційна безпека розглядається як стан системи, при якому :

1. Система здатна протистояти дестабілізуючій дії внутрішніх і зовнішніх загроз.

2. Функціонування і сам факт наявності системи не створюють загроз для зовнішнього середовища і для елементів самої системи.

На практиці інформаційна безпека зазвичай розглядається як сукупність наступних трьох базових властивостей інформації, що захищається, :

* конфіденційність, що означає, що доступ до інформації можуть отримати тільки легальні користувачі;
* цілісність, що забезпечує, що по-перше, інформація, що захищається, може бути змінена тільки законними користувачами, що мають відповідні повноваження, а по-друге, інформація внутрішньо-несуперечлива і (якщо ця властивість застосовна) показує реальний стан речей;
* доступність, що гарантує безперешкодний доступ до інформації, що захищається, для законних користувачів.

Діяльність, спрямовану на забезпечення інформаційної безпеки, прийнято називати захистом інформації.

Методи забезпечення інформаційної безпеки:

Теоретичні методи

* Формалізація процесів зв’язаних із забезпеченням ІБ.
* Обґрунтування коректності і адекватності систем забезпечення ІБ.

Організаційні методи

* Управління ІБ на підприємстві

Сервіси мережевої безпеки

* Ідентифікація і аутентифікація
* Розмежування доступу
* Протоколювання і аудит
* Засоби захисту периметра
* Криптографічні засоби захисту

Інженерно-технічні методи

* Захист інформації від витоку по технічних каналах

Правові методи

* Відповідальність
* Робота з державною таємницею
* Захист авторських прав
* Ліцензування і сертифікація

## 1.2. Поняття моделей безпеки комп'ютерних систем

Фундаментальним поняттям у сфері захисту інформації комп'ютерних систем є політика безпеки. Під нею розуміють інтегральну сукупність норм і правил, що регламентують процес обробки інформації, виконання яких забезпечує стан захищеності інформації в заданому просторі загроз. Формальне вираження політики безпеки (математичне, схемотехніка, алгоритмічне і т. д.) називають моделлю безпеки.

**Задачі, що вирішуються за допомогою моделей безпеки КС.** Моделі безпеки грають важливу роль в процесах розробки і дослідження захищених комп'ютерних систем, оскільки забезпечують системо-технічний підхід, що включає рішення наступних найважливіших завдань :

- вибір і обґрунтування базових принципів архітектури захищених комп'ютерних систем (КС), що визначають механізми реалізації засобів і методів захисту інформації;

- підтвердження властивостей (захищеності) систем, що розробляються, шляхом формального доказу дотримання політики безпеки (вимог, умов, критеріїв);

- складання формальної специфікації політики безпеки як найважливішої складової частини організаційного і документаційного забезпечення захищених комп'ютерних систем, що розробляються.

По суті моделі безпеки є початковим єднальним елементом в тріаді "Замовник (Споживач) -Розробник (Виробник) Експерт (Аудитор)". На основі моделей безпеки замовники можуть формулювати ті вимоги до захищених КС, які відповідають політиці безпеки, технологічним процесам обробки інформації, прийнятим у своїх організаціях і підприємствах. Розробники на основі моделей безпеки формують техніко-технологічні вимоги і програмно-технічні рішення по системах, що розробляються. Експерти, ґрунтуючись на моделях безпеки, будують методики і специфікації оцінки захищеності конкретних систем, здійснюють сертифікацію розроблених систем по вимогах захисту інформації.

**Основні положення суб'єктно - об'єктної формалізації комп'ютерних систем в аспекті захисту інформації.** Більшість моделей розмежування доступу ґрунтуються на представленні КС як сукупності суб'єктів і об'єктів доступу.

Приведемо основні положення суб'єктно-об'єктної формалізації комп'ютерних систем в аспекті безпеки інформації.

1. У КС діє дискретний час.
2. У кожен фіксований момент часу tk КС є кінцевою множиною елементів, що розділяються на дві підмножини:

* підмножина суб'єктів доступу S;
* підмножина об'єктів доступу O.

**Поняття суб'єкту та об'єкту комп'ютерної системи.** Під суб'єктом доступу розуміється активна суть КС, яка може змінювати стан системи через породження процесів над об'єктами, у тому числі, породжувати нові об'єкти і ініціалізувати породження нових суб'єктів.

Під об'єктом доступу розуміється пасивна суть КС, процеси над якою можуть в певних випадках бути джерелом породження нових суб'єктів

У моделі передбачається наявність апріорно безпомилкового механізму розрізнення активних і пасивних сутей (суб'єктів і об'єктів) по властивості активності, що можна проілюструвати інтуїтивно зрозумілими відмінностями між файлом з кодом програми і виконуваною(запущеною) програмою, що породжує процеси над об'єктами системи.

Крім того, передбачається також, що у будь-який момент часу tk, у тому числі і в початковий, множина суб'єктів доступу не порожня.

Користувачі КС представлені одним або деякою сукупністю суб'єктів доступу, діючих від імені конкретного користувача.

**Поняття користувача КС.** Під користувачем КС розуміється особа, зовнішній чинник, аутентифікований деякою інформацією, і управляючий одним або декількома суб'єктами, сприймаючий об'єкти та одержуючи інформацію про стан КС через суб'єкти, якими він управляє.

Таким чином, в суб'єктно-об'єктній моделі поняття суб'єктів доступу і користувачів не тотожні. Передбачається також, що призначені для користувача дії не можуть змінити властивостей самих суб'єктів доступу, що в загальному випадку не відповідає реальним КС, в яких користувачі можуть змінювати властивості суб'єктів, через зміну програм (виконуваних файлів). Проте подібна ідеалізація дозволяє побудувати чітку схему процесів і механізмів доступу, а загрози безпеки, що виникають внаслідок подібних реалій, розглядати в контексті гарантій виконання політики безпеки (політики розмежування доступу) через механізми незмінності властивостей КС.

**Створення суб' єктів КС іншими суб' єктами.** Суб'єкти КС можуть бути породжені з об'єктів тільки активною суттю (іншим суб'єктом).

Об'єкт оі називається джерелом для суб'єкта sm якщо існує суб'єкт sj, в результаті дії якого на об'єкт oi виникає суб'єкт sm

**Операція створення суб' єктів.** Відповідно, суб'єкт sj, називається активізуючим для суб'єкта sm .Для опису процесів породження суб'єктів доступу вводиться наступне позначення:

Create (sj, oi)→ sm - з об'єкту oi породжений суб'єкт sm при активізуючій дії суб'єкта sj Create називають операцією породження суб'єктів. Відмітимо також, що з огляду на те, що в КС діє дискретний час, то під впливом активізуючого суб'єкта у момент часу tk, новий суб'єкт породжується у момент часу tk + 1.

Результат операції Create залежить як від властивостей активізуючого суб'єкта, так і від властивостей об'єкту-джерела. Приміром, суб'єкт користувача у вигляді працюючого текстового редактора при відкритті файлу у форматі іншого текстового редактора може бути не здатним активізувати процедури обробки даних, що знаходяться там, а у кращому разі бути здатним тільки їх прочитати. Інший приклад - командний інтерпретатор ОС по команді користувача не може запустити на виконання текстовий файл і створити таким чином суб'єкт користувача. У таких випадках Create (sj, oi)→Ø.

**Асоціація об' єктів з суб' єктами.** Об'єкт oi у момент часу tk асоційований з суб'єктом sm, якщо стан об'єкту вплинув на стан суб'єкта в наступний момент часу tk+1 (тобто суб'єкт sm використовує інформацію, що міститься в об'єкті oi)

З визначення виходить, що об'єкт-джерело у момент породження суб'єкта є асоційованим з ним, а в наступні моменти часу може перестати або залишитися асоційованим з ним. Приміром, виконувані файли програм є асоційованими з суб'єктом тільки у момент його породження, оскільки в процесі ініціалізації (запуску) код програми з виконуваного файлу копіюється в спеціальну область пам'яті (сегмент коду), звідки згодом власне і витягаються команди-інструкції виконання програми. Отже, файл на диску з виконуваним кодом програми після її запуску перестає бути асоційованим з суб'єктом, породженим запуском програми. Навпаки, в деяких СУБД зі вбудованими системами програмування інтерпретаторського типу команди-інструкції по обробці даних в кожен момент часу можуть витягатися безпосередньо з файлів бази даних, таких, що розташовуються на диску. В цьому випадку, відповідно, файл бази даних продовжує залишатися асоційованим з суб'єктом, породженим відкриттям (запуском) відповідного файлу бази даних [3].

**Потоки інформації.** Активна суть суб'єктів доступу полягає в їх можливості здійснювати певні дії над об'єктами, що об'єктивно призводить до виникнення потоків інформації. Виходячи з цього, центральним положенням суб'єктно-об'єктної моделі є наступне.

Усі процеси безпеки в КС описуються доступами суб'єктів до об'єктів, що викликають потоки інформації.

Потоком інформації між об'єктом oi і об'єктом oj називається довільна операція над об'єктом oj, що реалізовується в суб'єкті sm і залежна від об'єкту oi .

Для опису потоків вводять наступне позначення:

Stream(sm, oi)→ oj - потік інформації від об'єкту oi(oj) до об'єкту oj(oi) в суб'єктові sm (через суб'єкт sm) [4].

Потік може здійснюватися у вигляді різних операцій над об'єктами - читання, зміна, видалення, створення і т. д. Об'єкти oi і oj, що беруть участь в потоці, можуть бути як джерелами, так і приймачами інформації, можуть бути як асоційованими з суб'єктом, так і неасоційованими, а також можуть бути порожніми (∅) об'єктами (наприклад, при створенні або видаленні файлів). Слід особливо підкреслити, потоки інформації можуть бути тільки між об'єктами, а не між суб'єктом і об'єктом, з причини того, що суб'єкт це активна суть, тобто дії, процеси і так далі, а інформація - пасивна суть, яка може розміщуватися, витягатися, породжуватися, змінюватися і так далі тільки в об'єктах. Активна роль суб'єкта полягає в самій реалізації потоку, в його локалізації в суб'єкті (через суб'єкт), у тому числі, через задіювання в потоці асоційованих з суб'єктом об'єктів (наприклад, буферів оперативної пам'яті). В цьому відношенні детальніший аналіз поняття суб'єктів доступу, показує, що асоційовані об'єкти можуть бути розділені на два види:

* функціонально-асоційовані об'єкти;
* асоційовані об'єкти-дані.

Функціонально-асоційовані об'єкти впливають (визначають) на самі процеси суб'єкта (наприклад, стан сегменту коду визначає властивості суб'єкта в наступний момент часу). Асоційовані об'єкти-дані виступають в ролі аргументів в операціях, що породжують потоки інформації (наприклад, буфери оперативної пам'яті, в яких поміщається для відображення на екрані інформація при читанні файлу). як образом, якщо на перший погляд в потоці бере участь тільки один (одні) суб'єкт(и), то, як правило, при пильнішому погляді можна побачити, що в цій операції беруть участь ще і асоційовані з суб'єктом доступу об’єкти [5].

**Поняття доступу.** Потік завжди ініціюється (породжується) суб'єктом доступу. На цій підставі вводиться наступне центральне в політиці і моделях розмежування доступу поняття.

Доступом суб'єкта sm до об'єкту oj називається породження суб'єктом sm потоку інформації між об'єктом oj і деяким(и) об'єктом oi (у т. ч., але не обов'язково, об'єкт oi асоційований з суб'єктом sm ).

Формальне поняття доступу дає можливість засобами суб'єктно-об'єктної моделі перейти безпосередньо до опису процесів безпеки інформації в захищених КС. З цією метою вводиться множина потоків P для усієї сукупності фіксованих декомпозицій КС на суб'єкти і об'єкти в усі моменти часу множина P є об'єднанням потоків по усіх моментах часу функціонування КС.

З точки зору процесів безпеки, що трактувала як стан захищеності інформації в КС, множина потоків P розбивається на дві підмножини PN і PL, що не перетинаються.

де PL - множина потоків, що викликаються легальними (безпечними) доступами;

PN - множина небезпечних, таких, що порушують стан захищеності інформації (конфіденційність, цілісність і доступність інформації) потоків в КС [5].

На основі множини потоків дається наступне поняття, що становить основу формалізації політики розмежування доступу в моделях безпеки.

Правилами розмежування доступу суб'єктів до об'єктів є формально описані потоки, що належать множині PL

Останнє визначення завершує основні положення суб'єктно-об'єктної моделі КС, на методологічному фундаменті якої будуються більшість моделей розмежування доступу, що виражають, власне, підходи, принципи і механізми правил розмежування доступу (політику розмежування доступу), а також формальні їх специфікації (самі моделі розмежування доступу). З огляду на те, що визначення не конкретизує і не деталізує конкретних механізмів фільтрації потоків на небезпечні і безпечні, то можна говорити, що суб'єктно-об'єктна модель КС інваріантна відносно будь-якої політики безпеки, що приймається в КС.

**Основні типи політики безпеки КС.** Методологічною основою для формування політик безпеки в захищених КС послужили реальні організаційно-технологічні схеми забезпечення безпеки інформації не в комп'ютерних сферах. Багато підходів до захисту комп'ютерної інформації було "підглянуто", зокрема, у сфері роботи з "паперовими" конфіденційними документами, простіше кажучи, у сфері діловодства.

Виділяється дві основних (базових) політики безпеки - дискреційна і мандатна. У ще не до кінця сталої термінології сфери захисту комп'ютерної інформації, першу називають політикою виборчого доступу, а другу - політикою повноважного доступу.

Слід зазначити, що відомі моделі ролевого доступу виділяють в групу особливої "ролевої політики безпеки". Крім того в документальних інформаційно-пошукових системах застосовується політика тематичного розмежування доступу. Моделі, що виражають ту або іншу політику безпеки, детально розглядаються у відповідних главах. Тут же ми обмежимося загальною їх характеристикою, відштовхуючись від основних понять і, зокрема, визначень суб'єктно-об'єктної моделі КС.

Політика дискреційного (виборчого) доступу. Множина безпечних (дозволених) доступів PL задається для іменованих користувачів (суб'єктів) і об'єктів явним чином у вигляді дискретного набору трійок "Користувач(суб'єкт) -поток(операція) - об'єкт".

Принцип дискреційної політики розмежування доступу можна охарактеризувати схемою "кожен-з-кожним", тобто іншими словами для будь-якої зі всіляких комбінацій "користувач (суб'єкт) ресурс (об'єкт)" має бути явно заданий дозвіл/заборона доступу і вид відповідної дозволеної/забороненої операції (Read, Write і т. д.). Таким чином, при дискреційній політиці розмежування доступу здійснюється найдетальнішим чином - до рівня окремо взятого суб'єкта, окремо взятого об'єкту доступу і окремо взятої операції [6].

Політика мандатного (повноважного) доступу. Множина безпечних (дозволених) доступів PL задається неявним чином через вступ для користувачів-суб'єктів деякої дискретної характеристики довіри (рівня допуску), а для об'єктів деякої дискретної характеристики конфіденційності (грифа секретності), і наділення на цій основі користувачів-суб'єктів деякими повноваженнями породжувати певні потоки залежно від співвідношення "рівень допуска - поток(операція) – рівень конфіденційності".

Таким чином, на відміну від дискреційної політики, при мандатній політиці розмежування доступу здійснюється менш детально - до рівня групи користувачів з певним рівнем допуску і групи об'єктів з певним рівнем конфіденційності. Помітимо також, що зменшення гранулюваності доступу створює умови для спрощення і поліпшення управління доступом зважаючи на істотне зменшення кількості суб'єктів управління і контролю.

Політика тематичного доступу. Множина безпечних (дозволених) доступів PL задається неявним чином через введення для користувачів-суб'єктів деякої тематичної характеристики - дозволених тематичних інформаційних рубрик, а для об'єктів аналогічної характеристики у вигляді набору тематичних рубрик, інформація по яких міститься в об'єкті, і наділ на цій основі суб'єктів-користувачів повноваженнями породжувати певні потоки залежно від співвідношення "набір тематичних рубрик суб'єкта-набір тематичних рубрик об'єкту"

Як і при мандатному доступі, тематичний принцип визначає доступ суб'єкта до об'єкту неявно, через співвідношення спеціальних характеристик суб'єкта, що пред'являються, і об'єкту і, відповідно, в порівнянні з дискреційним принципом істотно спрощує управління доступом.

Політика ролевого доступу. Множина безпечних (дозволених) доступів PL задається через введення в системі додаткових абстрактних сутей - ролей, виступаючих деякими "типовими" (ролевими) суб'єктами доступу, з яких асоціюється конкретні користувачі (в ролі яких здійснюють доступ), і наділення ролевих суб'єктів доступу на основі дискреційного або мандатного принципу доступу до об'єктів системи.

Ролева політика розмежовує доступ не на рівні користувачів-суб'єктів, а на рівні ролей, що є групами однотипного доступу до об'єктів системи, і на цій основі розвиває ту або іншу базову політику безпеки (дискреційну або мандатну). Тому в більшості джерел ролевий принцип розмежування доступом не виділяється в окрему політику, а розглядається як деякі доповнення до моделей дискреційного або мандатного доступу. Слід також відмітити, що в практиці функціонування захищених комп'ютерних систем широко використовується тимчасові і маршрутні (у розподілених КС) обмеження доступу, що дозволяє, в принципі, говорити про тимчасову і маршрутну політику безпеки, які, доповнюють відмічені базові політики безпеки. Кожна політика безпеки вимагає певної інформації для розмежування доступу в конкретній системі, що локалізується в об'єкті, що асоціюється з монітором безпеки. Для моделей дискреційного доступу ця інформація представляє список дозволених трійок "суб'єкт (користувач) - операція-об'єкт". Для управління доступом в системах з мандатним доступом потрібна інформація по рівнях допуску суб'єктів і грифах конфіденційності об'єктів. У системах ролевого доступу окрім інформації, що регламентує доступ ролей до об'єктів (на основі дискреційного або мандатного принципу), потрібна інформація за асоціацією користувачів-суб'єктів з ролями. При тематичному доступі потрібна інформація по тематичних рубриках користувачів-суб'єктів і об’єктів [7].

## 1.3. Представлення інформації в автоматизованій системі

Введемо деякі позначення Нехай A - кінцевий алфавіт, A\* - множина слів кінцевої довжини в алфавіті A, L ⊂ A\* - мова, тобто множина слів, виділених за певними правилами з A\*.

Будь-яка інформація в автоматизованій системі представляється словом в деякій мові L.

Назвемо об'єктом відносно мови L довільну кінцеву множина слів мови L. Очевидно, що як об'єкти можна розглядати багато сутей, що входять до складу АС. Наприклад, текстовий файл є об'єктом, оскільки в довільний момент часу в нього може бути записана деяка послідовність символів, яка в загальному випадку буде одним з кінцевої множини слів L над деяким алфавітом A. Аналогічно може бути складена мова, що описує клавіатуру і її стани в довільний момент часу. Мовою опису клавіатури можна вважати множину можливих її станів. Перетворенням інформації ми називатимемо відображення, задане на множині слів мови L. Іншими словами, перетворення відображує слово, що описує початкові дані, в інше слово. Помітимо, що сам опис перетворення при цьому також є словом. Прикладом перетворення може служити програма, написана на деякій мові програмування. Помітимо, що програма може або виконуватися, або просто зберігатися у файлі на деякому носію. Аналогічно, перетворення може:

* зберігатися - в цьому випадку опис перетворення зберігається в деякому об'єкті і нічим не відрізняється від інших даних [8];

- діяти - перетворення може взаємодіяти з деякими ресурсами АС.

Ресурси системи, виділені для дії перетворення, прийнято називати доменом. Щоб ініціювати дію перетворення, йому потрібно надати певний статус - передати управління. Перетворення, якому передано управління, називається процесом. У свою чергу, об'єкт, що описує перетворення, якому виділений домен і передано управління, називається суб'єктом.

## 1.4. Суб' єктно - об' єктне моделювання

**Реалізація доступу суб' єкта до об' єкта.** Суб'єкт для реалізації перетворення використовує інформацію, що міститься в об'єкті, тобто здійснює доступ до об'єкту. Існують два основні види доступу:

**Читання.** Якщо суб'єкт S дістає доступ до об'єкту O на читання, то це означає, що створюється перенесення інформації від об'єкту O до суб'єкта S - інакше кажучи

виникає інформаційний потік від O до S (рисунок 1.1).

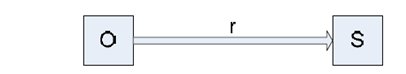


Рис 1.1. Інформаційний потік від O до S

**Запис.** Якщо суб'єкт S дістає доступ до об'єкту O на запис, то створюється перенесення інформації від суб'єкта S до об'єкту O, тобто виникає інформаційний потік від S до O (рисунок. 1.2).

Рисунок 7

Рис 1.2. Інформаційний потік від S до O

Дані два види доступу є базовими. Існують і складніші варіанти - наприклад, активізація процесу, коли суб'єкт S дістає доступ до об'єкту O на активізацію

процесу, записаного в O у вигляді даних. В цьому випадку для перетворення, описаного в O формується домен, і цьому перетворенню передається управління. Помітимо, що будь-який суб'єкт сам є об'єктом відносно деякої мови,то якщо S - множина усіх суб'єктів в системі, а O - множина усіх об'єктів, то S ⊆ O.

Сформульовані твердження дозволяють сформувати базову аксіому, що лежить в основі усієї формальної теорії захисту інформації:

Усі питання безпеки інформації описуються доступами суб'єктів до об'єктів.

**Монітор безпеки звернень.** Концепція монітора безпеки звернень є досить природною формалізацією деякого механізму, що реалізовує розмежування доступу в системі. Монітор безпеки звернень (МБЗ) є фільтром, який дозволяє або забороняє доступ, ґрунтуючись на встановлених в системі правилах розмежування доступу

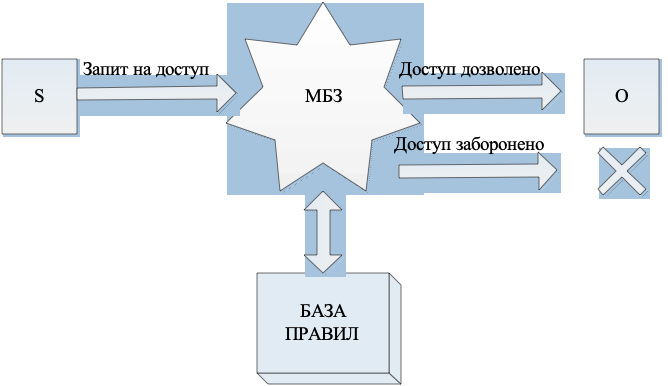


Рис 1.3. Монітор безпеки звернень

Отримавши запит на доступ від суб'єкта S до об'єкту O, монітор безпеки звернень аналізує базу правил, відповідну встановленій в системі політиці безпеки, і або дозволяє, або забороняє доступ.

Монітор безпеки звернень задовольняє наступним властивостям:

1. Жоден запит на доступ суб'єкта до об'єкту не повинен виконуватися в обхід МБЗ.

2. Робота МБЗ має бути захищена від стороннього втручання.

3. Представлення МБЗ має бути досить простим для можливості верифікації коректності його роботи.

### 1.4.1. Формальні моделі управління доступом.

**Модель Харрісона-Руззо-Ульмана.** Розроблена в 1971 р. модель Харрісона-Руззо-Ульмана формалізує матриці доступу - таблиці, що описує права доступу суб'єктів до об'єктів (рис. 2.4)

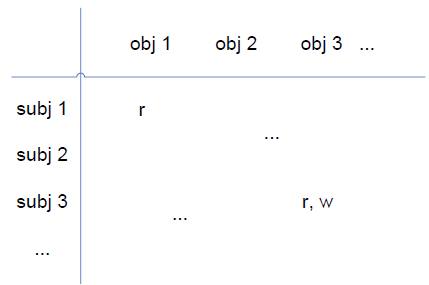


Рис 1.4. Матриця доступу

Рядки матриці доступу відповідають суб'єктам, існуючим системі, а стовпці - об'єктам. На перетині рядка і стовпця вказані права доступу відповідного суб'єкта до цього об'єкту: наприклад, на рисунку 1.4 суб'єкт subj 3 має права читання і запису по відношенню до об'єкту obj 3.

Введемо наступні позначення:

S - множина можливих субьектов

O - множина можливих обьктов

R - кінцева множина прав доступу

O×S×R - простір стану системи;

М - матриця прав доступу, що описує поточні права доступу суб’єктів до об’єктів;

Q=(S, O, M) - поточний стан системи;

M[s, o] - осередок матриці, що містить набір прав доступу суб'єкта s∈ S до

об'єкту o∈O.

Поведінка системи в часі моделюється переходами між різними її станами.

Модель передбачає наявність шести елементарних операцій :

1. enter r into M[s, o] ( s∈S, o∈O ) **-** додавання суб'єкту s права r по

відношенню до об'єкту o. В результаті виконання команди відбуваються

наступні зміни в стані системи

Помітимо, що вміст елементу таблиці розглядається як множина. Це, зокрема, означає, що якщо елемент, що додається, вже був присутній в осередку, то його вміст не змінюється.

1. delete r from M[s, o] ( s∈S, o∈O ) **-** видалення у суб’єкта s права r по

відношенню до об'єкту o. В результаті виконання команди відбуваються

наступні зміни в стані системи:

Якщо право, що видаляється, було відсутнє в осередку, то стан системи в результаті виконання цієї команди ніяк не зміниться.

1. create subject s (s∉S **) -** створення нового суб'єкта s. Зміни в стані системи:

Як бачимо, при створенні суб'єкта в матрицю M додаються рядок і стовпець.

1. destroy subject s (s∈S) **-** видалення існуючого суб'єкта s. Зміни в стані системи:
2. create object o (o∉O) **-** створення нового об'єкту o. Зміни в стані системи:
3. destroy object o (o∈O \ S) **-** видалення існуючого об'єкту o Зміни в стані системи:

Формальний опис системи в моделі Харрісона-Руззо-Ульмана виглядає таким чином. Система Σ = (Q, R, C) складається з наступних елементів:

1. Кінцевий набір прав доступу R={r1, ., rn}.

2. Кінцевий набір початкових суб'єктів S0={s1, ., sl}.

3. Кінцевий набір початкових об'єктів O0={o1, ., om}.

4. Початкова матриця доступу M0.

Поведінка системи в часі розглядається як послідовність станів {Qi}, кожен наступний стан є результатом застосування деякої команди до попереднього: Qn+1=Cn(Qn). Для заданої системи початковий стан Q0={S0, O0, M0} називається безпечним відносно права r, якщо не існує застосовної до Q0 послідовності команд, в результаті виконання якої право r буде занесено в осередок матриці M, в якій воно було відсутнє в стані Q0. Іншими словами це означає, що суб'єкт ніколи не отримає право доступу r до об'єкту, якщо він не мав його спочатку. Якщо ж право r виявилося в осередку матриці M, в якій воно спочатку було відсутнє, то говорять, що стався витік права r.

Нехай в початковому стані в системі є три суб'єкти: o, s і t; s - має право запису по відношенню до t, а t - має деяке право a (яке може бути або r, або w) по відношенню до o. Покажемо, як cуб’єкт s може отримати право доступу a по відношенню до суб'єкта o.

1. Система знаходиться в початковому стані.

2. Суб'єкт s створює новий суб'єкт x, по відношенню до якого автоматично отримує права читання і запису.

3. Суб'єкт s передає суб'єктові t права читання і запису по відношенню до x.

4. Суб'єкт t передає суб'єктові x право доступу a по відношенню до o.

5. Суб'єкт s отримує від суб'єкта x право доступу a по відношенню до o.

Приведена послідовність операцій проілюстрована на рисунку 1.5

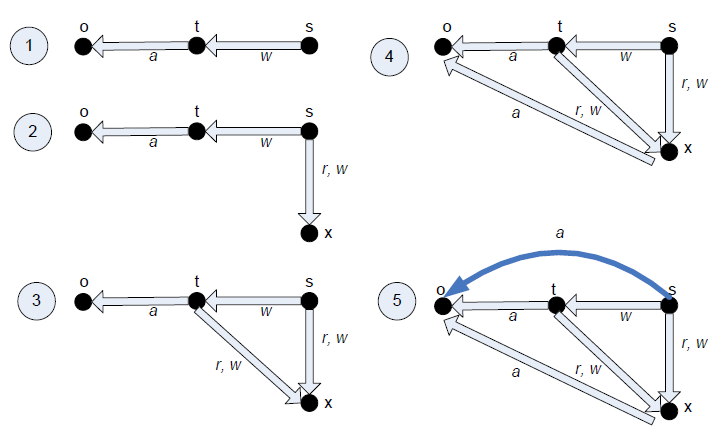


Рис.1.5. Витік права

Як бачимо в результаті виконання кроків 1-5, суб'єкт s обхідним шляхом отримує право доступу a по відношенню до суб'єкта o - тобто відбувається витік права a - а це означає, що початковий стан не є безпечним.

З практичної точки зору значний інтерес представляв би універсальний метод визначення того, чи являється задана система з деяким початковим станом безпечної відносно того або іншого права доступу.

### 1.4.2. Формальні моделі цілісності.

**Модель Кларка-Вілсона.** Модель цілісності Кларка-Вілсона була запропонована в 1987 р. як результат аналізу практики паперового документообігу, ефективної з точки зору забезпечення цілісності інформації. Модель Кларка-Вілсона є описовою і не містить яких би то не було строгих математичних конструкцій - швидше її доцільно розглядати як сукупність практичних рекомендацій по побудові системи забезпечення цілісності в АС.

Введемо наступні позначення:

- S - множина суб'єктів;

- D - множина даних в автоматизованій системі (множина об'єктів);

- CDI (Constrained Data Items) - дані, цілісність яких контролюється;

- UDI (Unconstrained Data Items) - дані, цілісність яких не контролюється;

При цьому D = CDI ∪UDI, CDI ∩UDI =Ø.

- TP (Transformation Procedure) - процедура перетворення, тобто компонент, який може ініціювати транзакцію - послідовність операцій, що переводять систему з одного стану в інший;

- - IVP (Integrity Verification Procedure) - процедура перевірки цілісності CDI.

Правила моделі Кларка-Вілсона :

1. У системі мають бути IVP, здатні підтвердити цілісність будь-якого CDI.

Прикладом IVP може служити механізм підрахунку контрольних сум.

2. Застосування будь-якої TP до будь-якому CDI повинне зберігати цілісність цього CDI.

3. Тільки TP можуть вносити зміни до CDI.

4. Суб'єкти можуть ініціювати тільки певні TP над певними CDI.

Ця вимога означає, що система повинна підтримувати стосунки виду (s, t, d), де s∈S, t∈TP, d∈CDI . Якщо відношення визначене, то суб'єкт s може застосувати перетворення t до об'єкту d.

5. Має бути забезпечена політика розподілу обов'язків суб'єктів - тобто суб'єкти не повинні змінювати CDI без залучення до операції інших суб'єктів системи.

6. Спеціальні TP можуть перетворювати UDI на CDI.

7. Кожне застосування TP повинне реєструватися в спеціальному CDI. При цьому:

- цей CDI має бути доступний тільки для додавання інформації;

- в цей CDI необхідно записувати інформацію, достатню для відновлення повної картини функціонування системи.

8. Система повинна розпізнавати суб'єкти, що намагаються ініціювати TP.

9. Система повинна дозволяти виробляти зміни в списках авторизації тільки спеціальним суб'єктам (наприклад, адміністраторам безпеки). Ця вимога означає, що трійки (s, t, d) можуть модифікувати тільки певні суб'єкти.

Безумовними достоїнствами моделі Кларка-Вілсона є її простота і легкість спільного використання з іншими моделями [3].

**Модель Біба.** Модель Біба була розроблена в 1977 році як модифікація моделі Белла- ЛаПадули, орієнтована на забезпечення цілісності даних. Аналогічно моделі Белла-ЛаПадули, модель Біба використовує решітку класів цілісності Λ = (IC,≤,-,⊗), де IC - класи цілісності даних. Базові правила Моделі Біба формулюються таким чином:

1. Просте правило цілісності (Simple Integrity, SI).

Суб'єкт з рівнем цілісності xs може читати інформацію з об'єкту з рівнем цілісності xo тоді і тільки тоді, коли xo переважає над xs.

2. \* - властивість (\* - integrity).

Суб'єкт з рівнем цілісності xs може писати інформацію в об'єкт з рівнем цілісності xo тоді і тільки тоді, коли xs переважає над xo. Для першого правила існує мнемонічне позначення No Read Down, а для другого - No Write Up.

Діаграма інформаційних потоків, відповідна реалізації моделі Біба в системі з двома рівнями секретності, приведена на рис. 1.7.

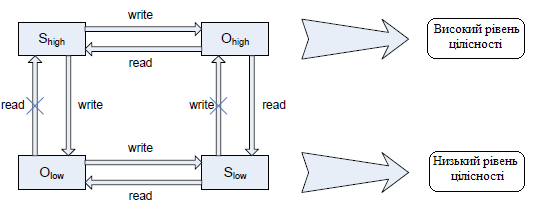


Рис. 1.7. Діаграма інформаційних потоків (модель Біба)

На окремий коментар заслуговує питання, що саме розуміється в моделі Біба під рівнями цілісності. Дійсно, в більшості випадків цілісність даних розглядається як деяка властивість, яка або зберігається, або не зберігається, - і введення ієрархічних рівнів цілісності може представлятися зайвим. Насправді рівні цілісності в моделі Біба варто розглядати як рівні достовірності, а відповідні інформаційні потоки - як передачу інформації з достовірнішої сукупності даних в менш достовірну і навпаки.

Формальний опис моделі Біба повністю аналогічно опису моделі Белла-ЛаПадули.

До достоїнств моделі Біба слід віднести її простоту, а також використання добре вивченого математичного апарату. В той же час модель зберігає усі недоліки, властиві моделі Белла-ЛаПадули.[3]

**Спільне використання моделей безпеки.** У реальних автоматизованих системах рідко зустрічаються системи захисту, орієнтовані виключно на забезпечення конфіденційності або виключно на забезпечення цілісності інформації. Як правило, система захисту повинна поєднувати обидва механізми - а значить, при побудові і аналізі цієї системи буде необхідним спільне використання декількох формальних моделей безпеки.

Розглянемо як приклад можливі варіанти спільного використання моделей Белла-ЛаПадули і Біба :

1. Дві моделі можуть бути реалізовані в системі незалежно один від одного. В цьому випадку суб'єктам і об'єктам незалежно привласнюються рівні секретності і рівні цілісності.
2. Можливе логічне об'єднання моделей за рахунок виділення загальних компонентів. У разі моделей Біба і Белла-ЛаПадули таким загальним компонентом є порядок розмежування доступу в межах одного рівня секретності

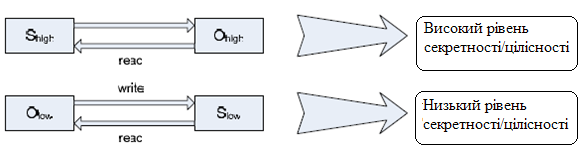
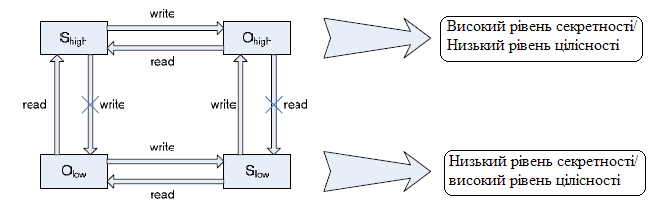


Рис 1.8 Спільне використання моделей Белла-ЛаПадули і Біба (виділення загальних компонентів)

3. Можливе використання одній і ой же решітка рівнів як для секретності, так і для цілісності. При цьому суб'єкти і об'єкти з високим рівнем цілісності розташовуватимуться на низьких рівнях секретності, а суб'єкти і об'єкти з низьким рівнем цілісності - на високих рівнях секретності.[7]

Рис 1.9 Спільне використання моделей Белла-ЛаПадули і Біба (єдина решітка рівнів цілісності і секретності)

Остання реалізація дозволяє, наприклад, розмістити системні файли на нижньому рівні ієрархії, що забезпечить їх максимальну цілісність, не акцентуючи увагу на зайвій в даному випадку секретності [9].

## 1.5. Висновки по розділу

В даному розділі були розглянуті основні положення в сфері захисту інформації. Виділено основні методи забезпечення інформаційної безпеки

. Визначено основні задачі, що вирішуються за допомогою моделей безпеки КС:

- вибір і обґрунтування базових принципів архітектури захищених комп'ютерних систем;

- підтвердження властивостей (захищеності) систем.

- складання формальної специфікації політики безпеки.

Більшість моделей розмежування доступу ґрунтуються на представленні КС як сукупності суб'єктів і об'єктів доступу, тому було введено суб'єктно - об'єктну формалізацію комп'ютерних систем. Визначено, що під суб'єктом доступу розуміється активна суть КС, а під об'єктом доступу розуміється пасивна суть КС, процеси над якою можуть в певних випадках бути джерелом породження нових суб'єктів. Користувачі КС представлені одним або деякою сукупністю суб'єктів доступу, діючих від імені конкретного користувача.

Виділено формальне поняття доступу, яке дає можливість засобами суб'єктно-об'єктної моделі перейти безпосередньо до опису процесів безпеки інформації в захищених КС.

Розглянуто основні поняття політики безпеки. Виділяється дві основних (базових) політики безпеки - дискреційна і мандатна.

Конкретна модель безпеки деталізує і формалізує (у вигляді аналітичних співвідношень, алгоритмів, і т. д.) загальний принцип розмежування доступу на основі однієї з розглянутих політик, а іноді деякій їх сукупності.

У конкретній КС розробники будують і реалізують оригінальні програмно-технічні рішення, що утілюють моделі безпеки, у тому числі структуру, функції, програмно-технічне втілення монітора безпеки.

Було розглянуто поширені моделі управління доступом, що дозволяють реалізувати формальний аналіз систем захисту, орієнтованих на забезпечення конфіденційності і цілісності інформації.

Моделі безпеки грають важливу роль в процесах розробки і дослідження захищених комп'ютерних систем, оскільки забезпечують системо-технічний підхід, що включає рішення наступних найважливіших завдань :

* вибір і обґрунтування базових принципів архітектури захищених комп'ютерних систем, що визначають механізми реалізації засобів і методів захисту інформації;
* підтвердження властивостей (захищеності) систем, що розробляються, шляхом формального доказу дотримання політики безпеки (вимог, умов, критеріїв);
* складання формальної специфікації політики безпеки як найважливішої складової частини організаційного і документаційного забезпечення захищених комп'ютерних систем, що розробляються

# Розділ 2. ТЕХНОЛОГІЯ БЛОКЧЕЙН, ЇЇ ПРИНЦИПИ РОБОТИ, ВИДИ, ПЕРЕВАГИ ТА НЕДОЛІКИ

## 2.1 Складові технології блокчейн

**Загальний опис технології.** Блокчейн - це структурований набір даних, що не змінюється Даними керує комп’ютерна система, що не належить жодному із її об’єктів. Всі дані розділяються на певні структуровані елементи - блоки. Кожен з них захищений і пов’язаний між собою за допомогою криптографічних принципів, які називаються “ланцюгом”. Однією з головних особливостей блокчейну є децентралізованість - тобто в системі відсутній центральний вузол, всі права, повноваження, обчислення розподілені рівномірно серед усіх об’єктів. Всі дані та операції з ними записуються в окрему так звану “книгу” - вона є спільною та незмінною, тому інформація в ній відкрита для всіх і кожного учасника системи. Отже кожен, хто бере участь у блокчейн-системі несе особисту відповідальність за свої дії.

Простою аналогією для розуміння є веб-сервіс для створення, редагування та збереження тестових документів Google Doc. При створенні документу і розповсюдженню його для певної групи людей - цей документ поширюється, замість того щоб копіюватись та передаватись. Цим самим створюється децентралізований ланцюг розповсюдження в певній системі зі своїми користувачами, який надає кожному з них доступ до документа одночасно. В цьому випадку всі модифікації документа записуються в режимі реального часу, що робить всі зміни абсолютно прозорими.

Цей кейс ілюструє три критичні ідеї технології блокчейн:

* Цифрові активи розповсюджуються, замість копіювання або передачі;
* Активи є децентралізованими, що забезпечує повний доступ в режимі реального часу;
* Прозора книга змін зберігає цілісність документу[10];

На рисунку 2.1 описана базова структура типової блокчейн технології:

* Користувач робить запит на запит інформації в блокчейн (транзакція)
* Транзакція потрапляє в P2p (peer-to-peer) систему, що складається з комп’ютерів (вузлів)
* Система вузлів верифікує (або не верифікує) транзакцію за допомогою одного з багатьох доступних алгоритмів (консенсусу)
* Як тільки транзакція верифікується - вона комбінується з іншими транзакціями - таким чином в ланцюг додається новий блок з даними
* Новий блок додано в існуючий блокчейн. Таким чином дані в системі залишаються постійними та незмінними

Рис. 2.1. Структура блокчейн технології

**Поняття та структура блоку.** Всі транзакції в системі блокчейн перманентно зберігаються в об’єктах, що називаються блоками. Блоки організовані в лінійну послідовність з плином часу. В різних блокчейн-системах структура блоку може відрізнятись, проте в загальному випадку складовими блоку вважаються:

* Час створення блоку (timestamp)
* Безпосередньо дані, що зберігаються (data)
* Хеш поточного блоку
* Хеш попереднього блоку

**Час створення блоку.** У кожного блоку є мітка часу, для точної ідентифікації моменту його створення. Для цього використовується поняття таймстемпу. Майже всі сучасні комп’ютерні системи оперують юнікс-таймстемпом. Це 13-значне ціле число, що дорівнює кількості мілісекунд, що минуло з моменту початку юнікс-епохи. Згідно стандарту ISO 8601 цією датою вважається 1 січня 1970 року 00:00:00:000 UTC (час за Гринвічем). Наприклад, якщо блок був створений о 10-ій годині ранку 10 лютого 2020 року - в блок буде записано значення 1549792800000.

**Дані.** При побудові блокчейн-систем виникає проблема зберігання інформації в блоці. Необхідно визначити який тип даних буде найінформативнішим для відображення інформації, та, водночас, буде практичним у використанні як для комп’ютерної системи, так і для звичайного користувача? Проаналізувавши основні формати передачі та зберігання даних, було визначено, що оптимальним буде застосування формату JSON.

Javascript Object Notation - це невеликий за обсягом формат обміну даними. Він легко сприймається людьми, а комп’ютери так само легко генерують та/або трансформують його для обробки.

У JSON-а присутні дві основні структури:

* Колекція пар ключ - значення У різних мова програмування це реалізується як об’єкт, запис, структура, словник, хеш-таблиця.
* Впорядкований список значень. В переважній більшості мов це масив, вектор, список або послідовність.

Це універсальні структури даних. Практично всі сучасні мови програмування підтримують їх у тій чи іншій формі [11].

**Хеш поточного та хеш попереднього блоку.** Одним із ключових елементів блоку є його хеш. Саме він є ключовим фактором захищеності блокчейн систем.

Значення хешу генерується на основі певної хеш-функції. Хеш-функція - це функція, що перетворює вхідні дані будь-якого розміру в дані фіксованого розміру.

У блоці вхідними даними для хеш-функції є:

* Таймстемп
* Безпосередньо дані блоку
* Хеш попереднього блоку

В сучасних блокчейн системах використовують такі хеш-функції: SHA-256, SHA-3, KECCAK-256 та інші. Їхніми перевагами є:

* швидка генерація вихідних даних
* менша ймовірність знаходження колізій відносно інших алгоритмів
* псевдовипадковість - при найменшій зміні вхідних даних - вихідні дані кардинально змінюються
* неможливо конвертувати хеш-функцію назад у вхідні дані

**Блок генезису.** Блок генезису є першим блоком у будь-якому протоколі на основі блокчейну, Це фундамент, з якого бере початок вся блокчейн система. На основі цього блоку послідовно додаються додаткові блоки, в результаті чого утворюється ланцюг блоків.

Блок генезису також називають “нульовим” блоком. Наступний блок, відповідно нумеруватиметься як блок номер один. Число, яке позначає порядковий номер блоку називається “висотою блоку”. Число висоти блоку є завжди натуральним числом.

Елементами блоку генезису можуть бути будь-які дані навіть в незашифрованому вигляді, оскільки за допомогою цієї інформації потенційний зловмисник не зможе скомпроментувати систему.

**Ланцюг та узгоджена зміна ланцюга.** Ланцюг блокчейну - це безпосередньо структурована система блоків. Цей ланцюг є загальнодоступним для всіх учасників системи і тому кожен має можливість відслідковувати кожен блок до найпершого генезисного блоку.

Валідація ланцюга - це концепція інспектування блокчейну та перевірки, що кожен блок був сконструйований коректно. Основними правилами, які забезпечують безвідмовну валідацію ланцюга є:

1. Блок повинен включати в себе всі необідні поля
2. Значення хешу попереднього блоку записане в поточному блоці має дорівнювати хешу попереднього блоку
3. Хеш повинен бути валідним. Потрібно перераховувати та валідувати хеш кожного блоку і перевіряти чи збігається він зі значенням хешу в блоці

Застосовуючи ці три правила до кожного блоку, можна констатувати що ланцюг є валідним.

Кожен суб’єкт в системі повинен мати певну відправну точку, тобто єдиний валідний ланцюг, від останнього блоку якого буде залежати генерація нового блоку. Саме для такого сценарію блокчейн ланцюг має властивість заміщення, іншими словами узгоджена зміна ланцюга. Для заміщення ланцюга, необхідне виконання двох умов:

1. Всі блоки ланцюга пройшли валідацію
2. Новий ланцюг містить в собі більше блоків ніж попередній.

Як результат відбувається заміщення - зміна попереднього набору блоків на новий валідний та більший за кількістю блоків ланцюг. На рис. 2.2. схематично показаний ланцюг, в якому блок жовтого кольору – це блок генезису, сірі блоки ті, що не пройшли валідацію, блоки блакитного кольору – ті, які безпосередньо формують ланцюг та пройшли валідацію.

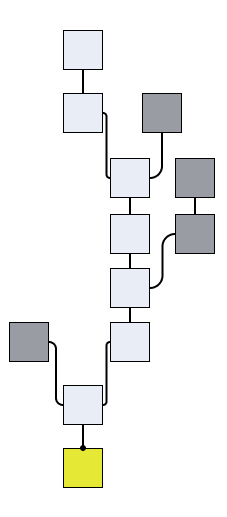


Рис. 2.2. Схема заміщення ланцюга

## 2.2 Основні принципи роботи технології блокчейн

З початком розвитку технології блокчейн, різні організації та ІТ-компанії розробили велику кількість програмних архітектур, які б відповідали різним бізнес, технічним та правовим критеріям. Враховуючи комплексність розроблених програмно-апаратних рішень, в роботі було виділено декілька принципів, які є фундаментальними для коректного функціонування технології блокчейн [12].

**Принцип децентралізованості.** Системи, базовані на технології блокчейн використовуються як надійно захищені від змін публічні репозиторії для документів, контрактів, записів та активів. Саме тому такі децентралізовані системи мають широкий спектр застосувань. До них, наприклад, належать смарт-контракти - певні дії між двома сторонами, які є автоматично-фіксованими та самовиконуваними, транзакції з декількома підписами - які вимагають присутності й виконання певних зобов’язань між багатьма суб’єктам, та право-власності - цифрове володіння матеріальними та нематеріальними активами, вбудованими в блокчейн, які можна відстежувати та обмінювати безпосередньо в самому блокчейні [13].

У цих випадках перевага блокчейну полягає в усуненні потреби довіреної третьої сторони (наприклад нотаріуса) та примусовому виконанні інструкцій за допомогою криптографічних засобів захисту. Таким чином, унеможливлюється шахрайство і знижується обсяг накладних витрат для учасників процесу.

Саме через такі переваги, пов’язані з автоматизацією, прозорістю та економічною ефективністю блокчейн, таким чином, може принести велику кількість критичних нововведень для багатьох різновидів контрактів та ділової діяльності.

Також блокчейн дає поштовх для розвитку нових систем управління з більш демократичним прийняттям спільних рішень та децентралізованих (автономних) організацій, які можуть самостійно оперувати з низьким рівнем впливу людського фактору.

Принцип децентралізованості не обмежується лише цим. Блокчейн може слугувати як розподілений, незворотній та зашифрований публічний засіб для голосування через мережу Інтернет. Виборці можуть самі верифікувати що їх голос був зарахований та, завдяки шифруванню, будь-яка система для голосування, що використовує блокчейн, не буде вразливою до взлому [14].

Використовуючи децентралізовані та зашифровані протоколи комунікації, інформація може зберігатись та розповсюджуватись серед великої кількості комп’ютерів, не зберігаючись на певних виділених серверах, контрольованими провайдером. Тим самим унеможливлюється керування цією системою певною особою, органом чи організацією [15].

Децентралізовані автономні організації можуть координувати свої дії між людьми та комп’ютерними системами за допомогою цифрових смарт-контрактів без необхідності традиційного керування суб’єктами господарювання

Всі вищенаведені приклади описують принцип децентралізованості як ключової особливості технології блокчейн.

**Принцип цілісності.** Для забезпечення цілісності даних в традиційних комп’ютерних системах, найчастіше використовують один з трьох методів зберігання інформації:

* метод зберігання WORM (Write once, read many) - один раз відбувається запис даних, з багаторазовим доступом до неї після цього
* використання певного сервісу для сертифікації даних (такі як цифровий нотаріус або нейтральна 3-тя сторона)
* складні системи контролю доступу

Але зараз організації звертають уваги на технологію блокчейн, який, як правило вважається захищеним від несанкціонованого доступу. [16]

За своїм задумом, блокчейн є стійким до модифікації даних. Блокчейн-книги є незмінними, при здійсненні транзакції або додавання даних в систему, вони не можуть бути зміненими чи видаленими. Крім того блокчейн це не лише структура даних, але й механізм зберігання часу для цієї структури тому історію зміни даних можна легко відновити задля подальшого використання [17].Одним із аспектів технології блокчейн, що є особливо важливим для поліпшення цілісності даних є дерево Меркла. Саме воно забезпечує цілісність даних в системі.

Дерево Меркла є фундаментальним компонентом технології блокчейн, що використовує криптографічні хеш-функції.

Кожен блок зберігає дані по транзакціям в формі дерева Меркла. Що є одним із видів структури даних. В ньому хеши блоків нижчих за порядком комбіновані в блоках вищих за порядком. Цей процес комбінації відбувається ітераційно аж до фінального блоку генезису. Цей блок являє собою відправною точкою для всього дерева, що містить всю інформацію.

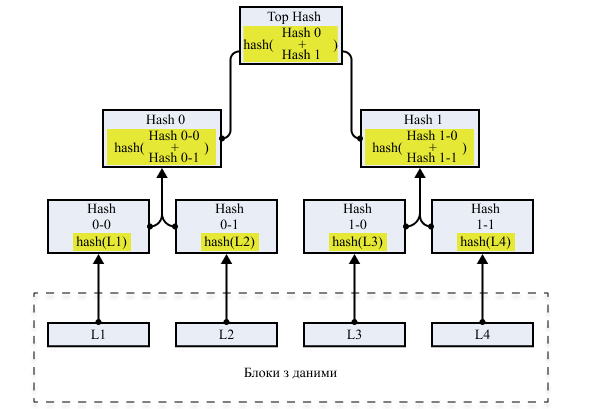
Найбільш поширеною формою дерева Меркла є бінарне дерево. На рис. 2.3. є 4 транзакції L1, L2, L3 та L4 в блоці. Кожна індивідуальна транзакція має свою хеш-функцію. Також, L1 має хеш 0-0, L2 - 0-1. Комбінації цих двох хешів зберігається в блоці, вищому на порядок. Тей самий алгоритм застосовується для L3, L4 та їхнього комбінованого блоку. Два блоки, вищі за порядком далі комбінуються в генезисний блок. Таким чином з’являється можливість відслідковувати всі транзакції аж до найпершого блоку, що забезпечує цілісність транзакцій, що зберігаються всередині блоку.

Рис. 2.3. Бінарне дерево Меркла

Варто зазначити, що технологія блокчейн може забезпечити цілісність збережених файлів в базі даних. Цього можна досягти завдяки добре сформованим транзакціям, аутентифікації та аудиту, які надає блокчейн. Завдяки цьому обсяг потенційних загроз цілісності даних може зменшитись [18].

**Принцип відкритості.** Всі записи підлягають аудиту заздалегідь обраним набором учасників, хоча набір може бути більш або менш відкритим. Наприклад, в публічних блокчейнах будь-хто з інтернет підключенням маючи доступ до мережі має рівні права до доступу та операцією над книгою. Тому всі записи є відкритими та такими, що легко відслідковуються. Більше того, учасники можуть здійснювати свої індивідуальні права (які, наприклад, вимірюються в обчислювальній потужності процесора) для оновлення книги. Учасники, також, мають можливість об’єднувати свої індивідуальні права [19].

**Принцип захищеності.** Блокчейни є спільними, захищеними від несанкціонованих дій книгами, де записи є незворотніми та не можуть бути підробленими завдяки криптографічним хеш-функціям. Хоча безпека є відносною концепцією, можна сказати що блокчейн є відносно безпечним, оскільки користувачі можуть передавати дані лише в тому випадку, якщо вони мають приватний ключ. Приватні ключі використовуються для створення підпису, що є обов’язковим компонентом кожної транзакції. Цей підпис, надалі, використовується для підтвердження того, що транзакція була отримана саме від цього користувача, а також, для запобіганню зміни транзакції піся того, як вона була створена.

## 2.3. Класифікація блокчейнів.

При побудові структури управління для блокчейну, важливо враховувати побудову книги. Залежно від її призначення, керування книгою може здійснюватись як і певним центральним органом так і децентралізовано, згідно з набором правил управління яких дотримуються та які застосовують всі учасники блокчейн-мережі. Структура управління визначає авторизацію та функції політики контролю. Це набір правил, якими керуються користувачі, система та її вузли для ініціювання певних заходів, пов’язаних з безпекою [19].

**За рівнем доступу.** Блокчейни можуть мати різні дозволи, відповідно до рівнів доступу до інформацію та управління нею. Саме відмінні риси повинні відповідати на наступні запитання:

* Які користувачі мають доступ до читання?
* Які користувачі мають доступ до запису?
* Чи є хтось, хто може “керувати консенсусом”? (тобто оновлювати і підтримувати цілісність книги)

Відповідно до набору правил управління, є різні конструкції системи, що дають відповіді на вищезазначені запитання, щоб краще служити приватним або державним інтересам, чи можливо загальним або спеціальним, унікальним інтересам. Хоча є безліч унікальних варіацій структури блокчейну, було виділено та класифіковано такі структури блокчейну за рівнем доступу:

1. Публічний блокчейн

У цьому випадку ні в кого немає переваги в керування або доступі до консенсусу. Усі учасники (вузли) мають доступ до читання/запису та без будь-якого централізованого контролю здійснювати операції над книгою. Прикладом може слугувати криптовалюта Біткоін. В ній кожен учасник може або обмінюватись біткоінами, або запустити вузол для участі в процесі перевірки транзакцій.

2. Публічний блокчейн з розмежуванням прав доступу

У такій моделі доступ на читання мають всі користувачі, але доступ на запис та/або керування консенсусом вимагає дозволу попередньо обраної групи вузлів. Прикладом такої моделі є система валових розрахунків в реальному часі - Ripple. Для валідації транзакції в цій системі, вузол має відноситись до обраної групи “Унікальний список вузлів”.

3. Приватний блокчейн з розмежуванням прав доступу

Для доступу до інформації в цій моделі, потрібно отримати дозвіл від централізованого органу. Такий підхід використовується переважно в корпоративних цілях, наприклад для відстеження доставки товарів. У випадку порушення процесу доставки є можливість дізнатись хто саме зі сторін зірвав поставку того чи іншого товару [19].

**За видом консенсусної мережі.** Топологія консенсусної мережі описує тип взаємозв’язків між вузлами і тип інформаційних потоків між ними для проведення транзакцій та/або з метою валідації. Децентралізовані рішення були представлені як альтернатива централізованим топологіям. З початку існування мережі Інтернет, системи поступово зміщуються в сторону децентралізованих механізмів, де всі вузли є еквівалентними. Мережі на основі блокчейну по суті створені за принципом однорангової мережі, яка в кінцевому підсумку визначає як відбуватиметься процес валідації.

Важливо зазначити, що топологія консенсусної мережі пов’язана з рівнем децентралізації, але не є єдиним визначальним фактором. Також інші фактори, такі як механізм винагороди сильно впливає на процес валідації.

Враховуючи вищесказане, можна розділити блокчейн мережі за видом консенсусної мережі на такі схеми:

1. Децентралізована

В таких мережах, як правило, створюється розподілена P2P мережа яка дозволяє здійснювати транзакції між кожним вузлом всередині мережі. Процес валідації в таких мережах відбувається майнерами та так званими супервузлами, які перевіряють транзакції всередині мережі, що підключена випадковим способом. Саме така топологія мережі ілюструє децентралізовану схему.

2. Ієрархічна

Є й інші, не децентралізовані схеми, а також існують відмінності між ролями, які набувають вузли. При такій топології, мережа розподілена на вузли відстеження та вузли валідації. Вузли відстеження - це шлюз для створення транзакції або виконання запитів у книгу. Вузли валідації мають приблизно ті ж самі функції як і вузли відстеження, проте вони також можуть сприяти створенню та запису додаткових записів в книгу. Такі топологічні схеми ще називають “Консорціум-блокчейнами”

3. Централізована

В деяких випадка, все ж необхідний один контролюючий орган для керування даними, що потрапляють до книги. Прикладом такої схеми є так звані “Цифрові Валюти Центрального Банку” (SBDC). В ньому використовуються додаткові механізми централізованої топології ,який по своїй суті є приватним блокчейном. Варто зазначити, що централізовані схеми не вважаються коректним дизайном з точки зору блокчейну [20].

**За типом консенсусу.** Загалом, толерантність до відмов в розподілених системах слід визначати стосовно трьох взаємопов’язаних питань:

* Несправність
* Помилки
* Зрив

Існує велика кількість різних типів помилок в системі, тому неможливо розробити ідеальну, безвідмовну та надійну систему. Блокчейн, як особливий вид розподілених систем, є толерантним до відмов, коли виявляє здатність продовжувати функціонувати. Тобто, він має забезпечувати достовірність, надійність та захищеність інформації, що зберігається в книзі [21]. Тому блокчейн-система потребує ефективного механізму консенсусу, для того щоб гарантувати що кожен вузол має свою оригінальну історію всіх транзакцій, яка буде консистентною та доступною всім іншим вузлам в мережі. Зважаючи на це, незмінність досягнутого консенсусу відрізняється, відповідно де ресурсів, необхідний для забезпечення безпеки великої мережі [22].

І дійсно, впродовж останніх років відбувається еволюція блокчейн технологій, що супроводжується розробкою різного роду механізмів, які гарантуються, що інформація буде надійною, захищеною то достовірною. За типом консенсусу, можна виділити наступні схеми:

1. Доказ роботи (Proof-of-Work)

Найбільш поширена криптовалюта в світі Біткоін використовує доказ роботи (Proof-of-Work - далі PoW) для забезпечення незмінності записів транзакцій. Обчислювальні пристрої (так звані майнери) під’єднанні до P2P мережі і виконують завдання перевірки транзакцій, запропонованих для додавання до існуючих, валідних транзакцій. Генерація блоку, для додавання його в мережу блокчейн вимагає інверсійного розшифрування критпографічної функції. Дешифрування можлива лише шляхом брутфорсу. В PoW ймовірність того, що майнер здобуде новий блок залежить від співвідношення між його обчислювальною потужністю та обчислювальною потужністю всі майнерів, підключених до мережі. Конкретно, манери повинні знайти рішення до односторонньою хеш-функції, обчислюючи нові хеш-значення, що базуються на комбінації попередніх хеш-значень, нової транзакції та одноразового коду. Розв’язок задачі є таким, що нове значення хешу повинне починатись з певної кількості нулів.

В такої схеми є і свої недоліки. Ці обчислення повинні відбуватись на дорогих комп’ютерних обчислювальних системах, що споживають багато енергії. За даним досліджень, майнинг криптовалюти Біткоін споживає 1 Гігават енергії щосекунди. Тому інші розробники блокчейн-систем намагаються розробити нові механізми для досягнення рівноцінного результату. Варто зазначити, що деякі криптовалюти використовують потужність POW для знаходження нових простих чисел або послідовності Кунінгама.

2. Доказ про частку (Proof-of-Stake) - далі PoS

В PoS генерація блоку тісно пов’язана з підтвердженням права власності на певну суму цифрових активів (наприклад цифрових валют), пов’язаних з блокчейн. Ймовірність того, що певний вузол буде обраним для верифікації наступного блоку прямо пропорційна до частки активів, яку вузол має в системі. Користувачі з більшою часткою багатства в системі, з великою ймовірністю будуть надавати достовірну інформацію стосовно процесу верифікації, і тому будуть вважати довіреними валідаторами. Було винайдено дві альтернативні методи PoS. Перший ґрунтується на випадковому виборі блоків. Обирається найменший хеш разом співставний з розміром частки. Цей процес детермінований, тому кожен вузол надалі може незалежно визначити ймовірність того, що його оберуть для верифікації наступної транзакції. Альтернативною схемою є вибір на основі віку, який комбінує випадковість віку монети (кількість активів вузла помножений на тривалість часу цих активів). Хоча PoS має шанс вирішити проблеми PoW (монополізаційні ризики та великий об’єм ресурсів для майнингу), PoS має декілька суперечливих моментів. Оскільки робота над декількома ланцюгами не потребує великої кількості ресурсів, можна зловживати системою, верифікуючи різні ланцюги, наслідком чого стане відсутність консенсусу. Цю проблему можна вирішити за допомогою делегованого доказу про частку. В цій схемі консенсусу транзакції валідують заздалегідь визначеним вузлами, що мають велику частку, набором користувачів. Ці користувачі нагороджуються за успішну валідацію, та караються за навмисне порушення.

3. Proof-of-Authority Підтвердження повноважень

В цьому випадку учасникам системи не потрібно вирішувати складні математичні ресурсоємкі завдання. Замість цього для валідації потрібно мати конфігураційний набір повноважень. Лише деяким вузлам дозволено створювати нові блоки та захищати блокчейн. Як правило, PoA механізми добре підходять для приватних мереж, де заздалегідь визначені користувачі з певними правами можуть контролювати дані, що надійдуть до публічного реєстру. Ці вузли повинні мати приватні ключі. Тому по кожному блоку (або його заголовку) що бачить клієнт може бути ідентифікований вузол, який створив цей блок. Проблеми PoA пов’язані з:

* Контролем частоти створення блоків
* Розподіл майнингового навантаження
* Підтримка динамічної зміни довірених користувачів

4. Proof-of-Capacity/Proof-of-Space and Proof-of-Storage Підтвердження обсягом пам’яті

PoC це реалізація популярної ідеї “об’єм пам’яті як ресурс”. Фокус іде не на потужність процесора а на обсяг фактичної енерго-незалежної пам’яті. Вузлам пропонується виділити значний обсяг їх місця на жорсткому диску для майнингу замість обчислювальної потужності як в PoW. Майнери мотивовані виділити місце на жорсткому диску, оскільки вузли з більшим обсягом вільної пам’яті мають більшу ймовірність успішно здобути блок і отримати за це винагороду. PoC використовує хеш дерева, для того щоб ефективно дозволити валідацію без зберігання всього дерева. Такі схеми є більш справедливими ніж PoW. Причина цього - менша варіативність часу доступу до пам’яті між вузлами та висока енергоефективність, завдяки відсутності обчислювальної потужності як ключового з компонентів.

PoC складається з взаємодії між доказовим вузлом P та вузлом-верифікатором V. P дає підтвердження для V що в нього є певна кількість вільних байтів. Після фази ініціалізації P повинен зберігати деякі дані F розміром N, V зберігає лише невелику кількість інформації. В будь-який інший період часу V може ініціювати фазу перевірки виконання і в кінці V генерує позитивний або негативний результат.

PoC визначається трьома величинами: N0, N1 i T. Майнер повинен довести, що або:

* Мав доступ до кількості пам’яті не меншої ніж N0 між фазами ініціалізації та перевірка і щонайменше N1 кількість пам’яті під час фази перевірки
* Або використав більше часу ніж T під час фази перевірки

У Proof of Storage механізм схожий до PoC але виділене місце використовується всіма учасниками як спільне хмарне сховище

5. Proof of Burn

В PoB майнери мають довести що вони “спалили” певну кількість цифрових активів. Це досягається відправкою активів (наприклад цифрових валют) верифікованому адресату, що співвідноситься з цим цифровим активом. По аналогії з PoS, PoB повинна мінімізувати витрату ресурсів, що генеруються через PoW. Але, на теперішній час, всі PoB механізми виконують роль “спалювачів” цифрових активів, згенерованими через PoW [23].

6. Гібридний

Існують і більш досконалі гібридні консенсуси. Зазвичай вони є комбінацією двох вищенаведених прикладів. Найбільш поширеними є “PoB i PoS” та “PoW і PoS”

## 2.4. Переваги та недоліки технології блокчейн

З тих пір, коли в 2009 році Сатоші Накамото представив світу технологію біткоін індустрія блокчейну виросла до великих масштабів. Популяризація блокчейну завдяки Біткоіну відкрила дорогу для дистрибутивних технологій. Реалізація блокчейну має великі перспективи [24]. Ідея, що стоїть за ним, досить проста, але потужна, оскільки, насамперед, складається з інноваційного підходу. Завдяки своїй модульності ця технологія має нескінченне різноманіття використання. Створення блокчейн технології виявило велику кількість переваг в різних сферах життя, забезпечуючи високий рівень безпеки в довірених мережах [25]. Проте, децентралізована природа також вносить деякі негативні та спірні моменти. В таблиці 2.1. було виділено та проаналізовано такі переваги та недоліки:

*Таблиця 2.1*

Переваги та недоліки технології блокчейн

|  |  |
| --- | --- |
| **Переваги** | **Недоліки** |
| Дані, що зберігаються в блокчейні є повноцінними, консистентними, точними та доступними | Велика кількість споживано енергії в окремих видах блокчейнів |
| Завдяки децентралізованим характеристикам, блокчейн краще протистоїть атакам та не має єдиної точки взлому | Верифікація сигнатури через криптографічні схеми займає багато часу та потребує великого обсягу обчислювальної потужності |
| Користувачі впевнені що транзакція буде виконана за певними правилами, відповідно відпадає необхідність в контролюючому органі | Загроза розділення ланцюга, коли вузол працює на застарілій версії програмного забезпечення а нова транзакція записується з вузла на новій версії |
| Зміни в публічному блокчейні є у відкритому доступі, тому всі дані є цілісними і не підлягають змінам | Дотримання балансу між кількістю вузлів та нагородою для користувачів |
| Оскільки всі дані зберігаються в одному публічному місці, відпадає потреба в підтримці великої кількості джерел даних  Висока швидкість проведення транзакцій | В деяких випадках складність обрання правильного механізму консенсусу |
| Надійна сумісність всіх даних | Недосконала правова база в провідних країнах світу |
| Невідворотність даних, що уже існують в мережі | В блокчейнах не забезпечується приватність даних, оскільки за своєю суттю всі дані доступні публічно |
| Відносно низька вартість за підтримку | При переході з більш класичних схем до блокчейну можуть виникати певні труднощі, пов’язані з ризиками безпеки та високою вартістю міграції |
| Можливість легко і просто віднайти в якому місці та в який період часу стався потенційний збій та відновити дані до правдивого вигляду | Інтеграція блокчейну в певну систему не може бути частковою, лише повноцінною |
| Стрімкий розвиток технологій та її високий потенціал | При невеликих операційних витратах, для блокчейну необхідні великі початкові інвестиції |
| Копії даних зберігаються та оновлюються на всіх вузлах системи | Можливість взлому публічних децентралізованих блокчейнів при наявності 51% всіє потужності мережі (атака 51%) |
| Застосування інноваційних підходів при розробці технологій | При розростанні ланцюга, кожному вузлу необхідно мати достатню кількість вільної пам’яті на жорсткому диску (в найбільшій блокчейн мережі Біткоін - ця цифра сягає 200 Гігабайт) |
| Велике різноманіття видів, типів блокчейну, механізмів консенсусів | Неможливість зміни даних після потрапляння її в мережу |
| Користувачі мають контроль над усією своєю інформацією та своїми транзакціями |  |
| Використання блокчейну як платформи для аналізу великої кількості даних |  |
| Низький поріг входу в систему |  |

## 2.5. Висновки по розділу

Блокчейн - це структурований набір даних, що не змінюється Даними керує комп’ютерна система, що не належить жодному із її об’єктів.

В даному розділі було розглянуто технологію блокчейн та виділено основні її складові.

Визначено поняття блоку та описано його компоненти.

* Час створення блоку (timestamp)
* Безпосередньо дані, що зберігаються (data)
* Хеш поточного блоку
* Хеш попереднього блоку

Проведено класифікацію блокчейнів за такими параметрами:

* За рівнем доступу
* За видом консенсусної мережі
* За типом консенсусу

Проаналізовано основні принципи технології блокчейн, такі як: принцип децентралізованості, принцип цілісності, принцип відкритості та принцип захищеності. Кожен з них грає ключову роль у функціонуванні технології блокчейн.

Розлянуто дерево Меркла в рамках криптозахисту блоків, де результати хеш функцій блоків напряму залежать один від одного і унеможливлюють підробку даних.

На основі проведених досліджень визначено основні, на думку автора, переваги та недоліки технології блокчейн.

# Розділ 3. РОЗРОБКА МОДЕЛІ КОМП’ЮТЕРНОЇ СИСТЕМИ НА БАЗІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

## 3.1 Аналіз моделі інформаційних потоків

Модель інформаційних потоків складається з об’єктів, зміни його станів та суб’єктів, що отримують доступ до об’єкту. Ця модель передбачає блокування несанкціонованого, незахищеного або обмеженого потоку інформації між суб’єктами та об’єктами на одному та/або різних рівнях класифікації.

В даній роботі прикладом для інтеграції системи блокчейн в модель безпеки комп’ютерної системи була обрана система відслідковування ланцюга фармацевтичних препаратів, оскільки саме на системах такого типу повністю розкривається потенціал технології блокчейн.

Ця система дозволяє кінцевому споживачу при отриманні фармацевтичного препарату простежити його шлях від заводу до аптеки для впевненості у якості товару.

Користувачі на заводах з виробництва препаратів вносять інформацію про препарат, далі препарат передається логістичним компаніям, які, у свою чергу, записують дані про доставку. Далі, препарат відправляється в аптеку, яка при його прийомі вносить інформацію про успішну доставку препарату. Наприкінці, особа, що придбала препарат, має можливість в аптеці перевірити всю інформації про товар, використавши комп’ютер, встановлений в залі самої аптеки.

Для побудови моделі безпеки системи відслідковування ланцюга доставки медичних препаратів необхідно:

* Визначити групи користувачів та види цінної інформації, до яких вони мають доступ
* Розробити веб-додаток, що включає в себе блокчейн смарт-контракт для реалізації суб’єктно-об’єктної моделі доступу та інтеграції блокчейну в модель безпеки
* Визначити програмно-апаратні засоби захисту для захисту комп’ютерної системи

**Групи користувачів та види цінної інформації, до яких вони мають доступ.** В системі існують 5 ролей користувачів:

1. Адміністратор

Користувач з роллю “Адміністратор” має повний спектр повноважень, що включає в себе:

* Створення та видалення користувачів
* Перегляд інформації про користувачів
* Перегляд інформації, що вносять користувачі

2. Завод

При виробництві препарату, користувач “Завод” на виробництві вносить інформацію про виготовлену партію фармацевтичного препарату

Такою інформацією є:

* Номер партії виробництва
* Дата виготовлення партії
* Строк придатності партії
* Назва товару

3. Логістична компанія

Товар передається логістичній компанії, яка зобов’язується доставити цю партію від заводу до аптеки. Користувач “Логістична компанія” вносить інформацію про доставку певної партії препарату в блокчейн

Перелік даних, що вносяться логістичною компанією:

* Номер партії виробництва
* Номер накладної
* Дата отримання товару від заводу
* Ідентифікатор аптеки-адресату товару

4. Аптека

Користувач “Аптека”, приймаючи товар від логістичної компанії, вносить інформацію про партію товару та його доставку

Дані:

* Номер партії виробництва
* Номер накладної
* Дата отримання партії

5. Покупець

Користувач “Покупець” отримує всю інформацію про препарат по конкретному ідентифікатору партії.

## 3.2 Розробка веб-платформи відслідковування ланцюга доставки медичних препаратів

Для реалізації проекту було обрано мову PHP для веб-застосунку та мову програмування Solidity для імплементації блокчейн смарт-контракту. На рисунку 3.1 схематично описана архітектура веб-платформи

Клієнт напряму взаємодіє з двома сутностями (рисунок):

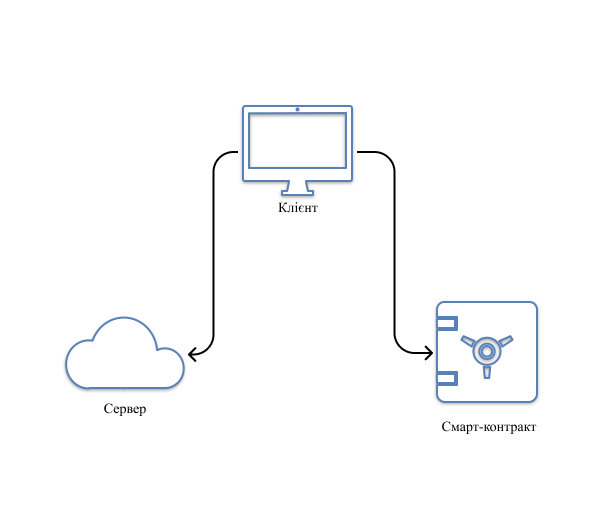
1. Сервер
2. Смарт контракт

Рис. 3.1. Схематична архітектура веб-платформи

Між клієнтом і сервером передаються дані про:

* Автентифікацію та авторизацію (логін, пароль, дані про сесію, доступ до певних ресурсів, в залежності від логіну та паролю)
* Інформація про користувачів в системі (адреса, описання, контактні дані, електронна пошта, номер телефону)

Клієнт та смарт-контракт обмінюються між собою такими даними:

* Публічний/приватний ключ користувача для доступу в блокчейн-систему
* Інформація про товар, його доставку логістичною компанією, та прийом аптекою

### 3.2.1 Імплементація блокчейн смарт-контракту.

Solidity - об’єктно-орієнтована високорівнева мова програмування створена для імплементації смарт-контрактів. Смарт-контракт - це алгоритм, що регулює поведінку акаунтів в криптосистемі Ethereum. Розробка велась в середовищі розробки Remix на базі тестової мережі Ropsten [26].

В смарт-контракті було створено 3 основних об’єкти:

1. Product - інформація про препарат, що створюється заводом
2. Invoice - накладна логістичної компанії
3. Information - кінцева інформація про препарат

**Об’єкт “Product”**

struct Product

{

string productName;

uint productionDate;

uint validUntilDate;

uint invoiceID;

uint ID;

}

**Об’єкт “Invoice”**

struct Invoice

{

Product [] products;

uint receivedFromFactory;

uint receivedFromLogistic;

address pharmacyID;

bool deliveryCompleted;

}

**Об’єкт “Information”**

struct Information

{

uint consignmentID;

string productName;

uint productionDate;

uint validUntilDate;

uint invoiceID;

uint receivedFromFactory;

uint receivedFromLogistic;

string addressTo;

}

Функції для додавання та видалення користувачів

**Функція створення користувача “Завод”**

function addFactory(address \_newFactory) onlyOwner public returns(bool)

{

require(!factories[\_newFactory] && !logistics[\_newFactory] && ! pharmacies[\_newFactory].exist);

factories[\_newFactory] = true;

return true;

}

**Функція видалення користувача “Завод”**

function deleteFactory(address \_newFactory) onlyOwner public returns(bool)

{

require(factories[\_newFactory]);

factories[\_newFactory] = false;

return true;

}

**Функція створення користувача “Логістична компанія”**

function addLogistic(address \_newLogistic) onlyOwner public returns(bool)

{

require(!factories[\_newLogistic] &&

!logistics[\_newLogistic] && !pharmacies[\_newLogistic].exist);

logistics[\_newLogistic] = true;

return true;

}

**Функція видалення користувача “Логістична компанія”**

function deleteLogistic(address \_newLogistic) onlyOwner public returns(bool)

{

require(factories[\_newLogistic]);

logistics[\_newLogistic] = false;

return true;

}

**Функція створення користувача “Аптека”**

function addPharmacy(address \_newPharmacy, string memory \_addressTo) onlyOwner public returns(bool)

{

require(!factories[\_newPharmacy] &&

!logistics[\_newPharmacy] && !pharmacies[\_newPharmacy].exist);

pharmacies[\_newPharmacy].exist = true;

pharmacies[\_newPharmacy].addressTo = \_addressTo;

return true;

}

**Функція видалення користувача “Аптека”**

function deletePharmacy(address \_newPharmacy) onlyOwner public returns(bool)

{

require(factories[\_newPharmacy]);

pharmacies[\_newPharmacy].exist = false;

return true;

}

**Додавання нового продукту користувачем “Завод”.** При додаванні інформації про продукт в блокчейн відбуваються такі дії:

1. Перевірка ролі користувача
2. Генерація транзакції для відправки її в блокчейн
3. Відправка транзакції. Комбінація її з іншими транзакціями для створення блоку
4. Перевірка валідності блоку майнерами криптосистеми Ethereum

Після виконання всіх вищезазначених кроків, інформація про медичний препарат заноситься в блокчейн. Формується такий масив даних для відправки в блокчейн:

[<<ідентифікатор\_партії>>, <<назва\_продукту>>, <<дата\_виготовлення>>, <<дата\_спожити\_до>>]

**Функція, для внесення інформації про препарат в блокчейн:**

function createProduct(

uint \_id,

string memory \_productName,

uint \_productionDate,

uint \_validUntilDate

) public returns(bool)

{

require (products[\_id].productionDate==0);

require(factories[msg.sender]);

products[\_id] = Product(\_productName, \_productionDate, \_validUntilDate, 0, \_id);

productsMas.push(\_id);

return true;

}

**Функція, для перегляду інформаціх про всі створені користувачем препарати:**

function getProducts() public view returns(uint [] memory)

{

return productsMas;

}  
На цьому етапі інформація про препарат виглядає наступним чином:

* “111111111111,test,1578865181,1581543580,0,0,0,”, де
* 111111111111 - ідентифікатор партії
* test - назва продукту
* 1578865181 - дата виготовлення
* 1581543580 - дата “Вжити до”

**Додавання інформації про доставку користувачем “Логістична компанія”.** Формування масиву даних для відправки в блокчейн:

[<<ідентифікатор\_партії>>, <<ідентифікатор\_накладної>>, <<публічний\_ключ\_гаманця\_аптеки>>]

Аналогічно з інформацією про препарат, створено функії для:

**1. Функція для внесення інформації про доставку в блокчейн:**

function transferProduct(uint \_id, uint \_invoiceID, address \_pharmacyID) public returns(bool)

{

require (!invoices[\_invoiceID].deliveryCompleted);

require(logistics[msg.sender]);

products[\_id].invoiceID = \_invoiceID;

invoices[\_invoiceID].products.push(products[\_id]);

invoicesMas.push(\_invoiceID);

invoices[\_invoiceID].receivedFromFactory = now;

invoices[\_invoiceID].pharmacyID = \_pharmacyID;

return true;

}

**2. Функція для перегляду інформації про всі доставки, що вніс користувач:**

function getInvoices() public view returns(uint [] memory)

{

return invoicesMas;

}

Після внесення даних про доставку, інформація про препарат матиме вигляд:

“111111111111,test,1578865181,1581543580,22222222,1578866339,0,Kyiv”, де:

* 22222222 - номер накладної логістичної компанії
* 1578866339 - дата прийняття препарату від заводу
* Kyiv - адреса аптеки, куди товар буде доставлений

**Додавання інформації про прийом товару користувачем “Аптека”.** Формування даних для відправки в блокчейн:

<<ідентифікатор\_накладної>>

Функція для внесення інформації про прийом товару в блокчейн

function receivedProduct(uint \_invoiceID) public returns(bool)

{

require (!invoices[\_invoiceID].deliveryCompleted);

require(pharmacies[msg.sender].exist);

invoices[\_invoiceID].receivedFromLogistic = now;

invoices[\_invoiceID].deliveryCompleted = true;

for(uint i=0; i<invoices[\_invoiceID].products.length; i++)

{

pharmacies[msg.sender].productsReceived

.push(invoices[\_invoiceID].products[i]);

}

return true;

}

**Перегляд повної інформації про товар.** Після внесення даних про прийом препарату аптекою, інформація про нього набуває кінцевого вигляду

“111111111111,test,1578865181,1581543580,22222222,1578866339,1578866903,Kyiv”, де

1578866903 - час прийому препарату аптекою

### 3.2.2 Імплементація веб-застосунку.

Для розробки веб-застосунку був використаний фреймворк Yii2 -  об’єктно-орієнтований компонентний фреймворк написаний на PHP, що реалізує парадигму Модель-Вид-Контроллер.

Для генерації користувацького інтерфейсу обрано інструмент Gii, що автоматично генерує контроллери та моделі відповідно до заданих параметрів.

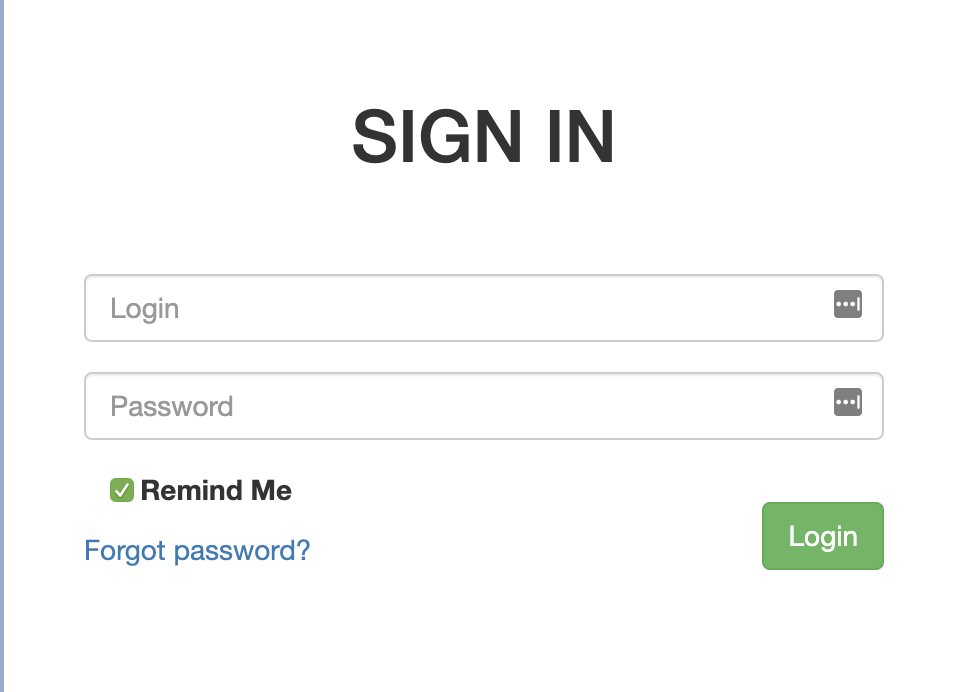
Першим кроком є аутентифікація та авторизація користувача з заданим логіном та паролем. На рисунку 3.2. показана форма логіну

Рис. 3.2. Форма логіну

За допомогою вбудованого в Yii2 authManager проводиться аутентифікація та авторизація юзера з відповідної роллю

public function actionLogin()

{

if (!Yii::$app->user->isGuest) {

return $this->goHome();

}

$model = new LoginForm();

if ($model->load(Yii::$app->request->post()) && $model->login()) {

$role\_name = key(Yii::$app->authManager->getRolesByUser(Yii::$app->user->identity->id))

if ($role\_name == 'admin')

return $this->redirect(['admin/index']);

else if ($role\_name == 'factory')

return $this->redirect(['factory/index']);

else if ($role\_name == 'logistics')

return $this->redirect(['logistics/index']);

else if ($role\_name == 'pharmacy')

return $this->redirect(['pharmacy/index']);

else if ($role\_name == 'user')

return $this->redirect(['user/index']);

else

return $this->goBack();

}

$model->password = '';

return $this->render('login', [

'model' => $model,

]);

}

**Користувач “Адміністратор”.** Адміністратор здійснює повний контроль над системою. В нього є можливість:

* Створювати та видаляти користувачів
* Переглядати інформацію про користувачів (рисунок 3.3)
* Переглядати інформацію, яку вносять користувачів (рисунок 3.4)



Рис. 3.4. Перегляд інформації про користувачів

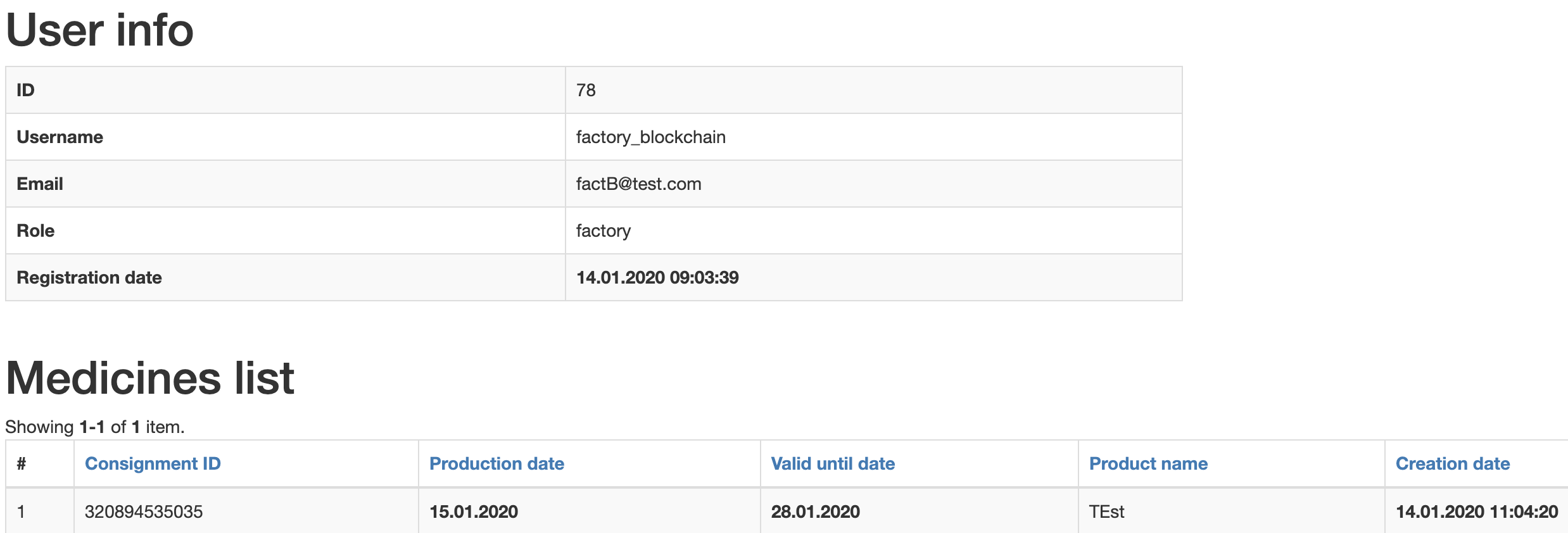


Рис. 3.5. Перегляд інформації, що вносять користувачі

Створення користувача показано на рисунку 3.6. та відбувається наступним чином:

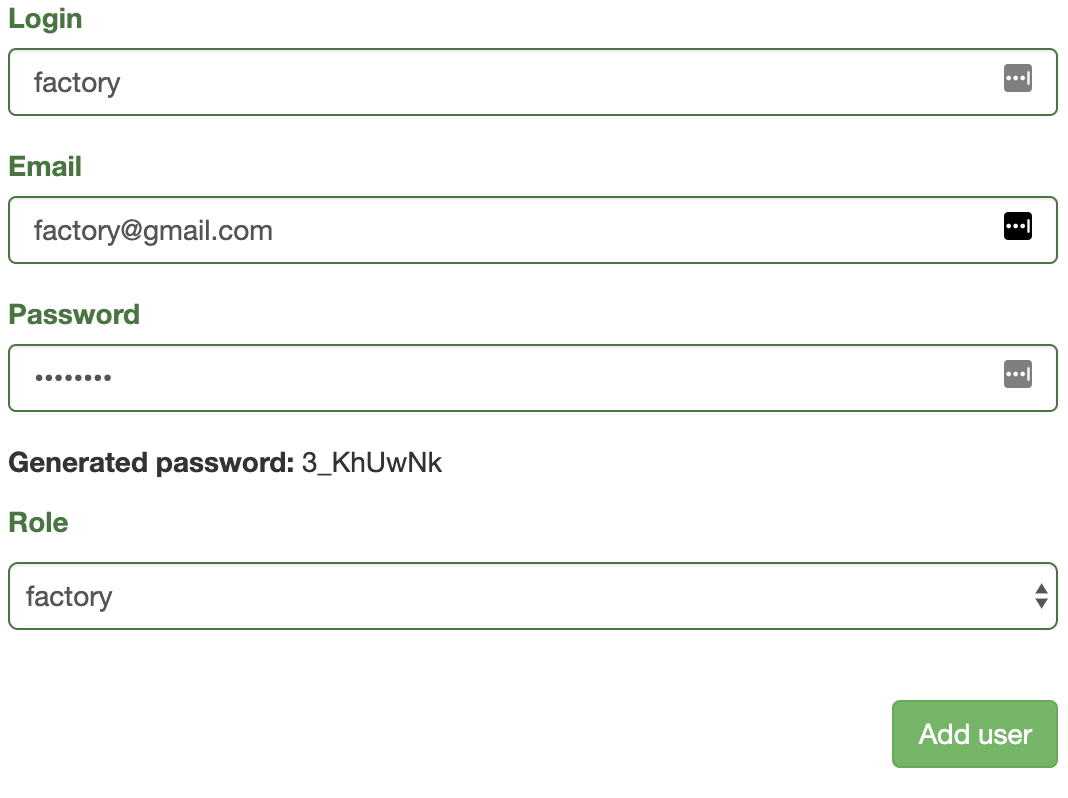
1. Адміністратор вводить логін, електронну пошту, пароль та роль юзера
2. Користувачу пошту відправляється створений логін та пароль
3. Інформація про користувача записується в базу даних на сервер
4. В блокчейні створюється гаманець користувача, що дозволяє підпису
5. вати та відправляти транзакції

Рис. 3.6. Форма створення нового користувача

**Функція для створення користувача:**

public function actionCreate() {

$model = $this->getSchemeUserModel();

$roles\_list = \Yii::$app->authManager->getRoles();

$roles\_list = array\_combine(array\_column($roles\_list,’name')

,array\_column($roles\_list,'name'));

if (Yii::$app->request->isPost && $model->load(Yii::$app->request->post())) {

$user = new User();

$user->username= Yii::$app->request->post('DynamicModel') ['username'];

$user->email= Yii::$app->request->post('DynamicModel')['email'];

$user->user\_role= Yii::$app->request->post('DynamicModel') ['user\_role'];

$user->setPassword(Yii::$app->request->post('DynamicModel') ['password']);

$user->generaneAuthKey();

}

**Створення гаманця користувача.** В блокчейн-системах використовують поняття “Гаманця” для ідентифікації користувача, що проводить транзакції в системі.

Гаманець складається з двох значень:

1. Публічний ключ. Віртуальна “адреса” користувача в криптосистемі Ethereum. Являє собою унікальний ідентифікатор гаманця
2. Приватний ключ - значення, за допомогою якого відбувається підпис будь-якої транзакції, здійсненої за допомогою цього гаманця.

Додатково, користувачу необхідно мати невелику кількість криптовалюти “Ефіру” для можливості здійснення транзакцій

Функція, що створює гаманець, зараховує певну кількість криптовалюти та повертає значення публічного та приватного ключа:

**Функція створення гаманця:**

public static function createWallet($privateKeyOwner) {

$config = [

'private\_key\_type' => OPENSSL\_KEYTYPE\_EC,

'curve\_name' => 'secp256k1'

];

$res = openssl\_pkey\_new($config);

if (!$res) {

echo 'ERROR: Fail to generate private key. -> ' . openssl\_error\_string();

exit;

}

// Generate Private Key

openssl\_pkey\_export($res, $priv\_key);

// Get The Public Key

$key\_detail = openssl\_pkey\_get\_details($res);

$pub\_key = $key\_detail["key"];

$priv\_pem = PEM::fromString($priv\_key);

// Convert to Elliptic Curve Private Key Format

$ec\_priv\_key = ECPrivateKey::fromPEM($priv\_pem);

// Then convert it to ASN1 Structure

$ec\_priv\_seq = $ec\_priv\_key->toASN1();

// Private Key & Public Key in HEX

$priv\_key\_hex = bin2hex($ec\_priv\_seq->at(1)->asOctetString()->

string());

$priv\_key\_len = strlen($priv\_key\_hex) / 2;

$pub\_key\_hex = bin2hex($ec\_priv\_seq->at(3)->asTagged()->asExplicit()- >asBitString()->string());

$pub\_key\_len = strlen($pub\_key\_hex) / 2;

// Derive the Ethereum Address from public key

// Every EC public key will always start with 0x04,

// we need to remove the leading 0x04 in order to hash it correctly

$pub\_key\_hex\_2 = substr($pub\_key\_hex, 2);

$pub\_key\_len\_2 = strlen($pub\_key\_hex\_2) / 2;

// Hashing

$hash = Keccak::hash(hex2bin($pub\_key\_hex\_2), 256);

// Ethereum address has 20 bytes length. (40 hex characters long)

// We only need the last 20 bytes as Ethereum address

$wallet\_address = '0x' . substr($hash, -40);

$wallet\_private\_key = $priv\_key\_hex;

self::sendEther($wallet\_address, 30000000000000, $privateKeyOwner); // 0.03eth

return [$wallet\_address, $wallet\_private\_key];

}

**Відправка електронного листа за допомогою вбудованого компоненту mailer**

$result = Yii::$app->mailer->compose()

->setFrom(['info@medblockchain.xyz'=>'Medblockchain'])

->setTo($user->email)

->setSubject('Registration on the website')

->setTextBody("Thank you for registering on the Medblockchain website."."\r\n\r\n".

"Here are your login details:"."\r\n\r\n".

"https://medblockchain.xyz/login"."\r\n\r\n".

Login: ".Yii::$app->request->post('DynamicModel') ['username']."\r\n\r\n".

"Password:".Yii::$app->request->post('DynamicModel') ['password']."\r\n\r\n".

"Regards, Medblockchain Team")

->setHtmlBody("Thank you for registering on the Medblockchain website."."<br><br>".

"Here are your login details:"."<br><br>".

"<strong>https://medblockchain.xyz/login</strong>"."<br><br>".

"Login: <strong>".Yii::$app->request->post('DynamicModel') ['username']."</strong><br><br>".

“Password: <strong>".Yii::$app->request->post('DynamicModel') ['password']."</strong><br><br>".

"Regards, Medblockchain Team") ->send();

**Функція перегляду користувачів та всієї їхньої інформації:**

public function actionView($id) {

$user = (new \yii\db\Query())

-> select('

{{%user}}.id,

{{%user}}.username,

{{%user}}.email,

{{%auth\_assignment}}.item\_name,

{{%auth\_assignment}}.created\_at')

-> from('{{%user}}')-> leftJoin(‘{{%auth\_assignment}}',

'{{%auth\_assignment}}.user\_id={{%user}}.id')

-> where(['=','id',$id])

-> one();

if ($user['item\_name'] == 'factory')

$products\_medicines = Products::find()->where(['user\_id'=>(int)$id]);

else if ($user['item\_name'] == 'logistics')

$products\_medicines = ProductsInfo::find()->where(['user\_id'=>(int) $id]);

else if ($user['item\_name'] == 'pharmacy')

$products\_medicines = AcceptedProducts::find()

->where(['user\_id'=>(int)$id]);

else

$products\_medicines = [];

$dataProvider = new ActiveDataProvider([

'query' => $products\_medicines,

'pagination' => [

'pageSize' => 20,

],

'sort' => [

'defaultOrder' => [

'id' => SORT\_ASC,

]

],

]);

return $this->render('view', [

'user' => $user,

'dataProvider' => $dataProvider

]);

}

**Функція для видалення користувача**

public function actionDelete($id) {

if (($model = User::findOne($id)) !== null) {

$model->delete();

Yii::$app->db->createCommand()

->delete(

'{{%auth\_assignment}}',

'user\_id = '.(int)$id)

->execute();

}

return $this->redirect(['admin/index']);

}

**Користувач “Завод”.** На рисунку 3.7. показано як користувач «Завод» вносить інформацію про:

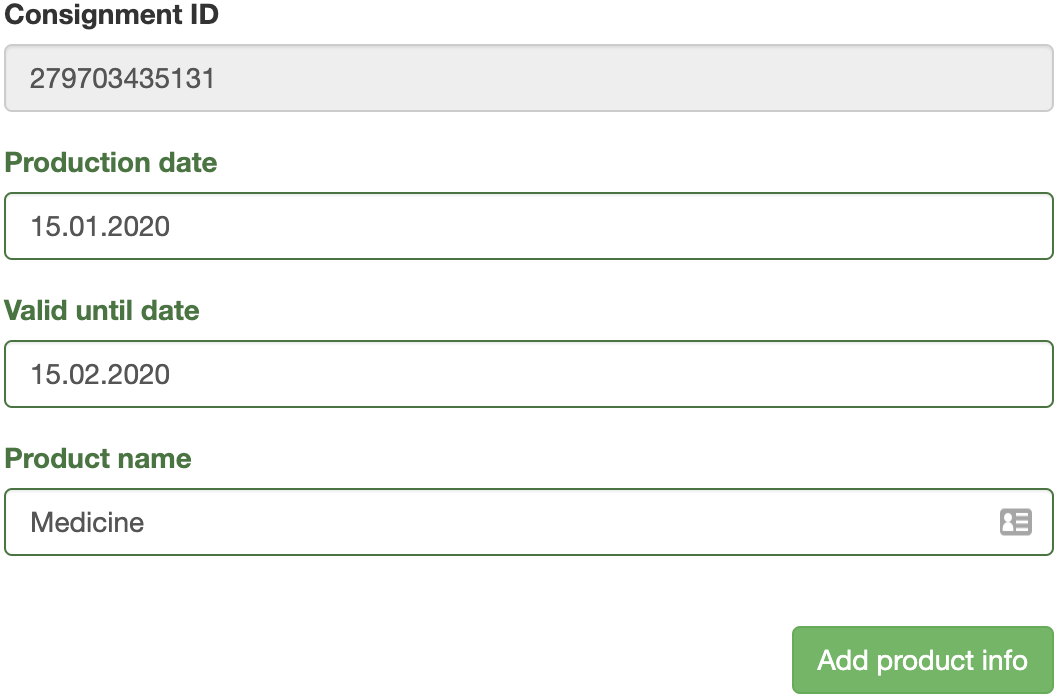
1. Номер партії
2. Дату виробництва
3. Дату “Вжити до”
4. Назву препарату

Рис. 3.7. Форма додавання інформації про препарат

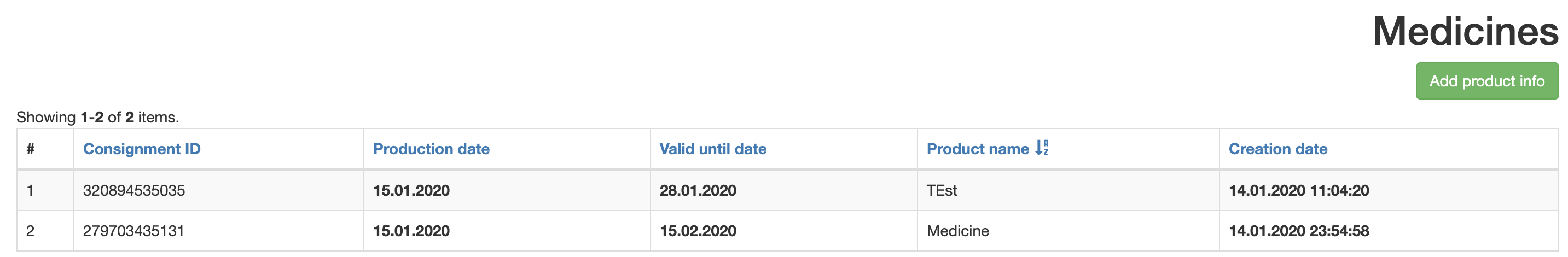
Після того як користувач заніс дані, він може їх переглянути. На рисунку 3.8. показана таблиця з внесеною користувачем інформацією.

Рис. 3.8. Перегляд створеної інформації користувачем

Відповідно викликається функція createProduct смарт-контракту для запису інформації в блокчейн.

public function actionAdd() {

$model = new Products();

if (Yii::$app->request->isPost && $model->load(Yii::$app->request- >post())) {

if ($model->production\_date)

$model->production\_date = date\_format(date\_create($model- >production\_date),'Y-m-d');

if ($model->valid\_until)

$model->valid\_until = date\_format(date\_create($model->valid\_until),'Y- m-d');

$model->user\_id = Yii::$app->user->identity->id;

if ($model->validate() && $model->save()) {

$user = User::find()

->where(['id'=>Yii::$app->user->identity->id])

->one();

ob\_start();

Medblockchain::createProduct($model->consignment\_id,$model- >product\_name,strtotime($model->production\_date),

strtotime($model->valid\_until),$user->wallet\_address,

$user->wallet\_private\_key);

$result = ob\_get\_contents();

ob\_end\_clean();

$transaction = new Transactions();

$transaction->name = 'Create product';

$transaction->transaction = $result;

$transaction->date\_ct = date\_format(date\_create(date()),'Y-m-d');

$transaction->save();

Yii::$app->session->setFlash('success', "Saved successfully!");

return $this->redirect(['factory/index']);

} else {

$model->production\_date = date\_format(date\_create($model- >production\_date),'d.m.Y');

$model->valid\_until = date\_format(date\_create($model- >valid\_until),'d.m.Y');

Yii::$app->session->setFlash('error', "Error!");

}

}

**Користувач “Логістична компанія”.** На рисунку 3.9. показана форма, за допомогою якої користувач “Логістична компанія" вносить інформацію про:

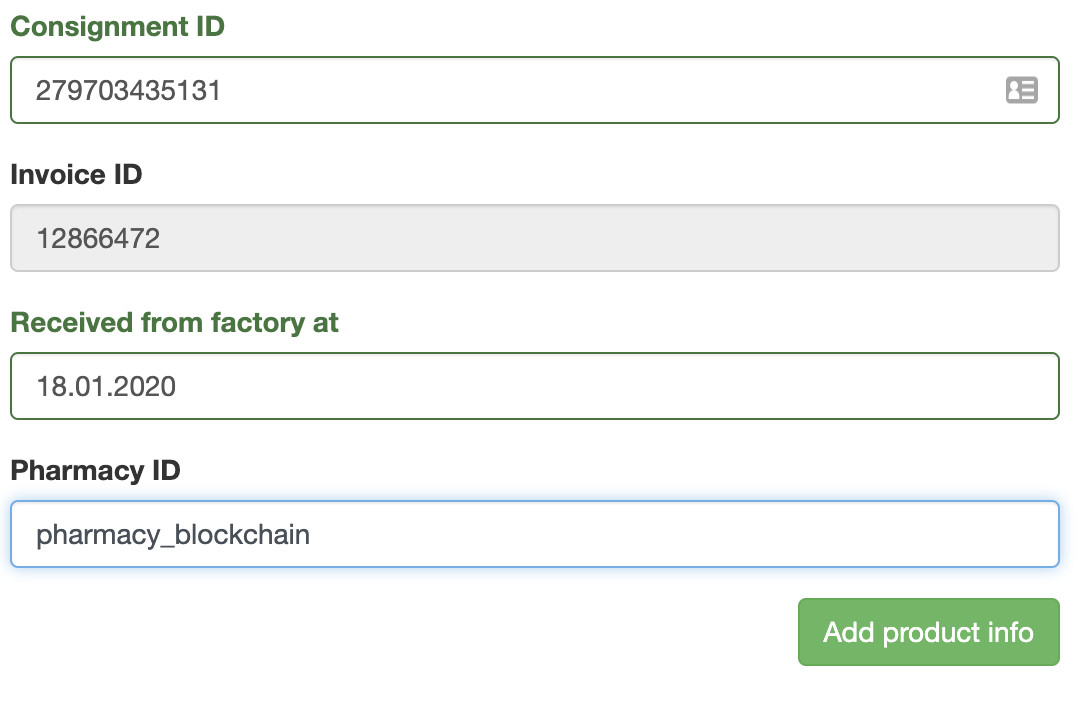
1. Номер партії
2. Номер накладної
3. Дату отримання препарату від заводу
4. Ідентифікатор аптеки-адресату

Рис. 3.9. Внесення інформації про доставку

Відповідно викликається функція transferProduct смарт-контракту для запису інформації в блокчейн

Після внесення відповідних даних, користувач має можливість переглянути їх в таблиці (рисунок 3.10)

Рис. 3.10. Перегляд внесеної інформації

public function actionAdd() {

$model = new ProductsInfo();

if (Yii::$app->request->isPost && $model->load(Yii::$app->request- >post())) {

$model2 = Products::find()->where(['consignment\_id'=>$model- >consignment\_id])->one();

if ($model->received\_from\_factory\_at)

$model->received\_from\_factory\_at = date\_format(date\_create($model- >received\_from\_factory\_at),'Y-m-d');

$model->user\_id = Yii::$app->user->identity->id;

if ($model->validate() && $model->save()) {

$user = User::find()

->where(['id'=>Yii::$app->user->identity->id])

->one();

ob\_start();

Medblockchain::transferProduct($model2->consignment\_id,$model- >invoice\_id,$model->pharmacy\_id,$user->wallet\_address,$user- >wallet\_private\_key);

$result = ob\_get\_contents();

ob\_end\_clean();

$transaction = new Transactions();

$transaction->name = 'Transfer product';

$transaction->transaction = $result;

$transaction->date\_ct = date\_format(date\_create(date()),'Y-m-d');

$transaction->save();

Yii::$app->session->setFlash('success', "Saved successfully!");

return $this->redirect(['logistics/index'])}

else {

$model->received\_from\_factory\_at = date\_format(date\_create($model- >received\_from\_factory\_at),'d.m.Y');

Yii::$app->session->setFlash('error', "Error!");

}

}

**Користувач “Аптека”.** На рисунку 3.11. показана форма, за допомогою якої користувач “Аптека" вносить інформацію про:

1. Номер партії
2. Номер накладної
3. Дату отримання препарату від логістичної компанії

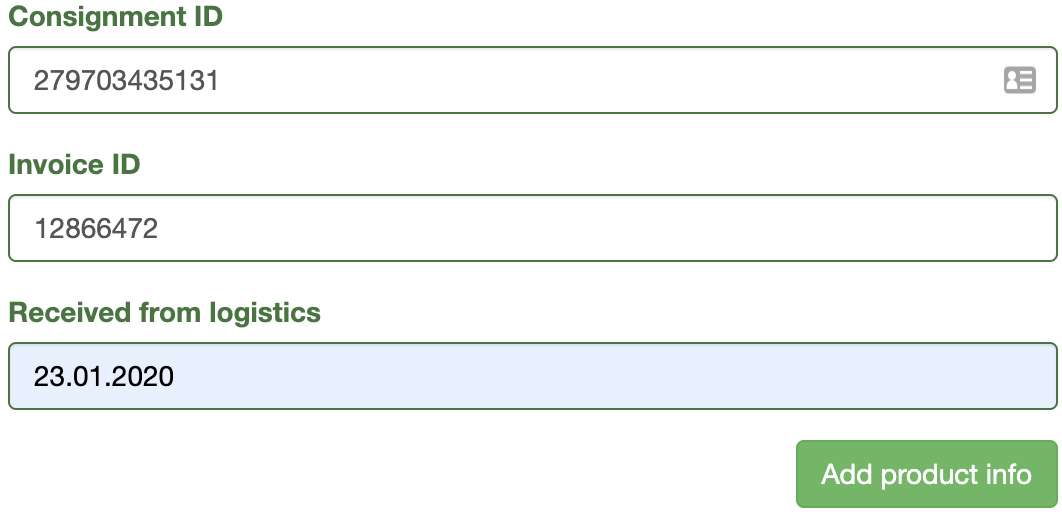
Відповідно викликається функція receivedProduct смарт-контракту для запису інформації в блокчейн

Рис. 3.11. Внесення інформації про доставку

Після внесення відповідних даних, користувач має можливість переглянути їх в таблиці (рисунок 3.12)

Рис. 3.12. Перегляд внесеної інформації

#### pasted-image.png

public function actionAdd() {

$model = new AcceptedProducts();

if(Yii::$app->request->isPost && $model->load(Yii::$app->request- >post())) {

$model2 = ProductsInfo::find()->where(['consignment\_id'=>$model- >consignment\_id])->one();

if ($model->received\_from\_pharmacy\_at)

$model->received\_from\_pharmacy\_at = date\_format(date\_create($model->received\_from\_pharmacy\_at),

'Y-m-d');

$model->user\_id = Yii::$app->user->identity->id;

if ($model->validate() && $model->save()) {

$user = User::find()

->where(['id'=>Yii::$app->user->identity->id])

->one();

ob\_start();

Medblockchain::receivedProduct($model2->invoice\_id,$user- >wallet\_address,$user->wallet\_private\_key);

$result = ob\_get\_contents();

ob\_end\_clean();

$transaction = new Transactions();

$transaction->name = 'Received product';

$transaction->transaction = $result;

$transaction->date\_ct = date\_format(date\_create(date()),'Y-m-d');

$transaction->save();

Yii::$app->session->setFlash('success', "Saved successfully!");

return $this->redirect(['pharmacy/index']);

} else {

$model->received\_from\_pharmacy\_at = date\_format(date\_create($model->received\_from\_pharmacy\_at),

'd.m.Y');

Yii::$app->session->setFlash('error', "Error!");

}

**Користувач “Покупець”.** Користувач “Покупець” отримує всю інформацію про препарат по конкретному ідентифікатору партії. На рисунку 3.13 показаний результат пошуку по певному номеру партії

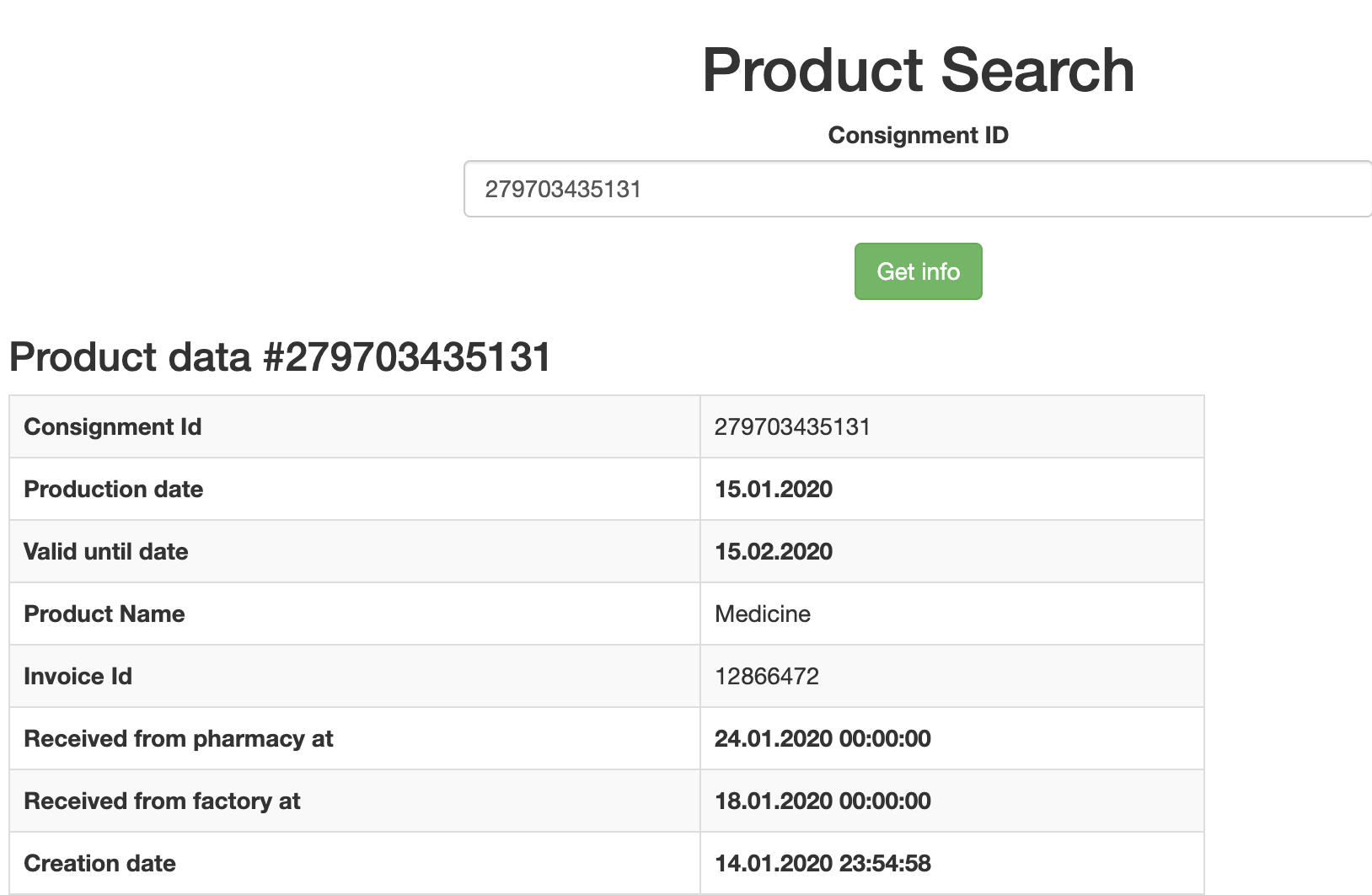


Рис. 3.13. Результат пошуку по певному номеру партії

public function actionIndex() {

$model = $this->getSchemeSearchConsignmentModel();

if (Yii::$app->request->isPost && $model->load(Yii::$app->request- >post())) {

$search\_model = Products::find()

->select('

{{%products}}.consignment\_id AS id,

{{%products}}.consignment\_id AS consignment\_id,

{{%products}}.production\_date AS production\_date,

{%products}}.valid\_until AS valid\_until,

{{%products}}.product\_name AS product\_name,

{{%products\_info}}.invoice\_id AS invoice\_id,

{{%accepted\_products}}.received\_from\_pharmacy\_at AS received\_from\_pharmacy\_at,

{{%products\_info}}.received\_from\_factory\_at AS received\_from\_factory\_at,

{{%products}}.date\_ct AS date\_ct,')

->leftJoin('{{%products\_info}}', ‘{{%products\_info}}

.consignment\_id={{%products}}.consignment\_id')

->leftJoin(‘{{%accepted\_products}}',

‘{{%accepted\_products}}

.consignment\_id={{%products}}.consignment\_id')

->where(['{{%products}}.consignment\_id'=>$model->consignment\_id])

->asArray()

->one();

} else {

$search\_model = new Products();

}

## 3.4 Оцінка захищеності комп’ютерної системи

В криптосистемі Ethereum при додаванні нового блоку, майнер повинен порахувати хеш-функцію для цього блоку, використвуючі такі дані:

1. Хеш попереднього блоку
2. Дані поточного блоку
3. Число “nonce” (number that only can be used once - число що може використовуватись лише одного разу)

Більше того, хеш-функція повинна відповідати критерію “складності” - кількості нулів на початку хеш-функції. Чим більше нулів - тим важче віднайти потрібний хеш. Майнеру потрібно перераховувати хеш блоку, що змінюючи значення nonce, поки хеш не відповідатиме критерію складності. Цей процес називається Proof-of-Work (доказ роботи). Над його виконанням трудяться майнингові ферми, що складаються з великої кількості програмно-апаратних комплексів з високою обчислювальною потужністю.

Всі вищеописані кроки застосовують для додавання нового блоку в блокчейн. Проте, які дії необхідно здійснити зловмиснику для зміни інформації в існуючому блоці? Оскільки всі блоки пов’язані між собою своїми хеш-значеннями, йому потрібно перерахувати кожний блок, починаючи з цільового блоку атаки і до блоку, що був доданий в блокчейн останнім. Тільки виконавши такі маніпуляції, зловмисник зможе переконати всіх учасників системи у валідності його ланцюга. На момент написання цієї роботи, криптосистема Ethereum має такі показники:

* Складність дорівнює 2093 [TH](https://etherscan.io/chart/difficulty) (середньо-статистична кількість спроб обчислення хешу одного блоку).
* Хеш-рейт складає 166521 Гігахешів за одну секунду (здатність системою приймати на перевірку лише певну кількість хешів за одну секунду).
* Новий блок додається в систему в середньому один раз за 30 секунд.

<https://etherscan.io/>

Тому, зловмиснику, для успішної атаки на крипто-систему Ethereum шляхом зміни існуючих даних необхідно:

1. Мати в розпорядженні надвеликий об’єм обчислювальних потужностей, для обчислення хешу нових блоків швидше ніж новий блок буде доданий в систему
2. На противагу цьому, не перевищити загальний ліміт хеш-рейту.
3. Оскільки складність блоку змінюється динамічно, базуючись на кількість обчислених блоків в системі, зі збільшенням кількості заново аномально-перерахованих блоків, складність буде збільшуватись.

Враховуючи всі ці фактори можна вважати що запропонована модель безпеки комп’ютерної системи на базі технології блокчейн є досить захищеною від спотворення даних.

## 3.5 Висновки по розділу

В даному розділі було розроблено модель безпеки комп’ютерної системи відслідковування ланцюга фармацевтичних препаратів на базі технології блокчейн.

Визначено групи користувачів та види цінної інформації, до яких вони мають доступ.

Створено веб-додаток, що включає в себе блокчейн смарт-контракт для реалізації суб’єктно-об’єктної моделі доступу та інтеграції блокчейну в модель безпеки.

Розроблено смарт-контракт на мові програмування Solidity. Реалізовано функції для додавання інформації про товар в блокчейн, створення та видалення гаманців користувачів в крипто-системі Ethereum.

Імплементовано веб-частину на базі фреймворку Yii2 та Cii. Створено сторінку авторизації, особистий кабінет для користувачів. Користувачі мають можливість вносити дані в систему за допомогою форм а також переглядати внесені ними дані. Панель адміністратора дозволяє створювати і видаляти користувачів, переглядати записи що вони вносять.

Встановлений зв’язок між функціями веб частини та аналогічними функціями смарт-контракту.

Оцінено захищеність комп’ютерної системи відносно можливості спотворення даних.

# ВИСНОВКИ

На сьогоднішній день питання забезпечення безпеки комп’ютерних систем є ключовим в наскрізь пронизаним IT-технологіями світі. Щороку зловмисники знаходять вразливості в класичних засобах захисту. Насамперед, технологія блокчейн здійснила революцію у взаємовідносинах користувачів та сторонніх інстанцій завдяки властивості децентралізованості. Проте блокчейн здатен зробити великий внесок в розвиток безпеки інформації.

Результатом виконаної роботи є вирішення задачі побудови моделі безпеки комп’ютерної системи на базі технології блокчейн.

У процесі виконання роботи отримані наступні результати:

1. Проаналізовано існуючі моделі безпеки, розглянуто основні поняття що лежать в основі моделей. Проведено дослідження моделей управління доступом та цілісності, визначено їхні принципи роботи.
2. Розглянуто технологію блокчейн. Визначено її складові та принципи роботи. Проведено класифікацію різних видів блокчейнів. Виявлено основні переваги та недоліки, що дало основу для подальших досліджень в даній області.
3. Розроблено модель безпеки комп’ютерної системи на базі технології блокчейн. Визначено основні групи користувачів ти види цінної інформації до яких вони мають доступ. На основі цього розроблений смарт-контракт на мові програмування Solidity та веб-додаток для реалізації суб’єктно-об’єктної моделі.

Проведені дослідження, практична реалізація веб-додатку та смарт-контракту а також додаткові програмно-апаратні засоби вказують на високий рівень захищеності комп’ютерної системи згідно критеріїв захищеності НД ТЗІ 2.5-004-99.

Рекомендовано використовувати дану модель безпеки комп’ютерної системи для платформи відслідковування ланцюга доставки медичних препаратів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. World Wide Web: Вартість витоку даних в 2019 році [Електронний ресурс] – Режим доступу <https://databreachcalculator.mybluemix.net/executive-summary> .

2.World Wide Web: Україна спільно з Bitfury Group переведуть державні реєстри на платформу блокчейн [Електронний ресурс] – Режим доступу <http://dir.gov.ua/2017/04/17/%D1%83%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D0%B0-%D1%81%D0%BF%D1%96%D0%BB%D1%8C%D0%BD%D0%BE-%D0%B7-bitfury-group-%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%B5%D0%B4%D1%83%D1%82%D1%8C-%D0%B4%D0%B5%D1%80%D0%B6%D1%80/> .

3. Дєвянін П.Н. Модели безопасности компьютерних систем: Учеб. пособие. – М.: Изд.центр «Академия», 2005. – 144 с.

4. Ed Tittel, JamesM. Stewart, Mike Chapple. CISSP: Certified Informations Systems Security Professional. Study guide. 2-nd Edition. – Sybex, 2003.

5. Гайдамакін Н.А. Разграничение доступа к информации в компьютерних системах. - Екатеринбург: изд-во Урал. Ун-та, 2003. – 328 с.

6. Корт С.С. Теоретические основи защити информации: Учебное пособие. – М.: Гелиос АРВ, 2004. – 240 с.

7. Щербаков А.Ю. Введение в теорию и практику компьютерной безопасности. М.: издатель Молгачев С.В.- 2001- 352 с.

8. А.А. Грушо, Є.Є. Тимоніна. Теоретические основы защиты информации. – М.:«Яхтсмен», 1996.

9. Д.П. Зегжда, А.М. Ивашко. Основы безопасности информационных систем. –М.: Горячая линия – Телеком, 2000.

10. World Wide Web: What is a Blockchain technology [Електронний ресурс] - Режим доступу <https://builtin.com/blockchain>.

11. World Wide Web: Introducing JSON [Електронний ресурс] – Режим доступу <http://www.json.org/>.

12. Atzori, Marcella. (2017). Blockchain technology and decentralized governance: Is the state still necessary?. Journal of Governance and Regulation.

13. Aaron Wright, Primavera De Filippi “Decentralized Blockchain Technology and the Rise of Lex Cryptographia” (2015)

14. World Wide Web: Helios Voting [Електронний ресурс] – Режим доступу <https://vote.heliosvoting.org/>

15. World Wide Web: Telehash secure mesh protocol (v3) [Електронний ресурс] – Режим доступу <https://telehashorg.readthedocs.io/en/latest/>

16. World Wide Web: How to Build Data Integrity into Blockchain Applications [Електронний ресурс] – Режим доступу <https://www.devprojournal.com/technology-trends/blockchain/how-to-build-data-integrity-into-blockchain-applications/>

17. World Wide Web: Blockchain & Data integrity [Електронний ресурс] – Режим доступу <https://medium.com/@hmishfer17/blockchain-data-integrity-e70e17cac086>

18 I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev and L. Yalansky, "Ensuring data integrity using blockchain technology," 2017 20th Conference of Open Innovations Association (FRUCT), St. Petersburg, 2017, pp. 534-539.

19. Tasca, Paolo, Tessone, Claudio. (2019). A Taxonomy of Blockchain Technologies: Principles of Identification and Classification. Ledger.

20. K. Wüst, A. Gervais, "Do you Need a Blockchain?," 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), Zug, 2018

21. Y. S. Han, R. Zheng, Wai Ho Mow, "Exact regenerating codes for Byzantine fault tolerance in distributed storage," 2012 Proceedings IEEE INFOCOM, Orlando, FL, 2012

22. Fischer M.J. (1983) The consensus problem in unreliable distributed systems (a brief survey). In: Karpinski M. (eds) Foundations of Computation Theory. FCT 1983. Lecture Notes in Computer Science, vol 158. Springer, Berlin, Heidelberg

23. Sunoo Park, Albert Kwon, Georg Fuchsbauer SpaceMint: A Cryptocurrency Based on Proofs of Space (2015)

24. World Wide Web: ADVANTAGES & DISADVANTAGES OF BLOCKCHAIN TECHNOLOGY [Електронний ресурс] – Режим доступу <https://blockchaintechnologycom.wordpress.com/2016/11/21/advantages-disadvantages/>

25. World Wide Web: Overview of Current Blockchain Applications: Advantages and Disadvantages [Електронний ресурс] – Режим доступу <https://applicature.com/blog/blockchain-startups/overview-of-current-blockchain-applications-advantages-and-disadvantages>

26. World Wide Web: Solidity 0.6.2 documentation [Електронний ресурс] – Режим доступу <https://solidity.readthedocs.io/en/latest/>

27 . Термінологія в галузі захисту інформації в комп’ютерних системах від несанкціонованого доступу [Текст]: НД ТЗІ 1.1-003 – 1999. – Чин. 1999. 04.28. – К.: ДСТСЗІ СБ України, 1999. – 12 с.