

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра
(ступінь вищої освіти)

на тему Розробка захищеного хмарного сховища для підприємства «Комтел»
(назва теми)

Виконав: студент 4 курсу, групи КНТ-519

Спеціальності _____
123 «Комп'ютерна інженерія»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія

ХРИСТЕНКО М.В.
(ПРІЗВИЩЕ та ініціали)

Керівник КАСЬЯН М.М.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЄФИМЕНКО М.В.
(ПРІЗВИЩЕ та ініціали)

(назва освітньої програми (спеціалізації))

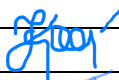
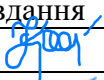
« » 2023 року

(ПРИЗВИЩЕ, ім'я, по батькові)

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

Пл4 – Основні етапи роботи системи:

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
	КАСЬЯН М.М., к. т. н., доцент		
Нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання «8» травня 2023року.

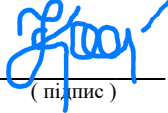
КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
	Аналіз технічного завдання	1 тиждень	виконано
	Аналіз структури роботи	1 тиждень	виконано
	Вибір відповідної технології для створення системи	2 тиждень	виконано
	Підбір комплектуючих елементів для створення системи, схеми підключення	2 тиждень	виконано
	Написання програмного коду	3 тиждень	виконано
	Реалізація проєкту	3 тиждень	виконано
	Оформлення проєкту	4 тиждень	виконано
	Проходження нормоконтролю	4 тиждень	виконано
	Проектування програмних засобів	4 тиждень	виконано
	Захист роботи	4 тиждень	виконано

Студент(ка)

 **Михайло ХРИСТЕНКО**
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

 **Микола КАСЬЯН**
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 64 с., 10 рис., 5 табл., 16 джерел.

СХОВИЩЕ ДАНИХ, ІНФОРМАЦІЙНА СИСТЕМА, "ХМАРА", КОРПОРАТИВНЕ СХОВИЩЕ, ЗАХИЩЕНЕ ХМАРНЕ СХОВИЩЕ, ІДЕНТИФІКАЦІЯ, АУТЕНТИФІКАЦІЯ, КРИПТОГРАФІЧНІ МЕТОДИ, ШИФРУВАННЯ, ДЕШИФРУВАННЯ.

Актуальність цієї роботи полягає в тому, що зберігання інформації на електронних носіях присутнє в кожній організації незалежно від її розмірів. Незважаючи на постійний прогрес у сфері зберігання інформації, поточний рівень розвитку корпоративних сховищ не задовольняє висунутим до них вимогам. Ці вимоги формуються в результаті зміни законодавства в галузі зберігання інформації та пов'язані, в основному, із забезпеченням інформаційної безпеки. Крім цього, користувачі, які працюють із корпоративними інформаційними системами, хочуть від таких систем зручності та простоти у використанні, як під час роботи з персональними сховищами інформації, що використовуються в приватних цілях.

У роботі розглянуто наявні способи організації сховищ на підприємствах, виявлено їхні переваги та недоліки. Також розглянуто наявні корпоративні хмарні рішення. На основі проведеного аналізу запропоновано структуру захищеного хмарного сховища для системи інформування підприємства «Комтел».

У результаті роботи було представлено проєкт організації хмарного сховища, застосування якого в інформаційній системі підприємства «Комтел» дасть змогу спростити роботу користувачів, які безпосередньо працюють зі сховищем, а також підвищити рівень інформаційної безпеки.

ЗМІСТ

Вступ.....	6
1 Аналіз технологій зберігання інформації.....	8
1.1 Існуючі підходи до зберігання інформації.....	8
1.2 Технології для створення систем зберігання даних	9
1.3 Зберігання інформації з використанням баз даних речей	13
1.4 Хмарні технології	15
1.5 Приклади хмарних рішень зберігання корпоративної інформації	20
1.6 Проблеми захисту сховищ інформації	23
1.7 Технології захисту хмарних сховищ.....	26
2 Організація інформаційної системи	29
2.1 Зберігання інформації на підприємствах	29
2.2 Основні характеристики хмарного сховища підприємства	32
2.3 Клієнтський доступ до сховища	33
2.4 Сховище для інформаційної системи.....	36
2.5 Забезпечення захисту сховища системі інформування	41
3 Організація захищеного хмарного сховища	46
3.1 Структура та організація сховища.....	46
3.2 Програма для модератора системи інформування.....	48
3.3 Модуль програми для клієнта-термінала	57
Висновки	61
Перелік джерел посилання	62

Перелік графічного матеріалу:

Пл1 – Інформаційні ресурси підприємства в корпоративній мережі

Пл2 – Завантаження файлу в сховище

Пл3 – Ієрархічна структура сховища;

Пл4 – Основні етапи роботи системи;

ВСТУП

Використання інформації в цифровій формі стало невід'ємною частиною будь-якої організації незалежно від її розмірів. Зберігання інформації на електронних носіях є досить зручним, оскільки вона легко структурується порівняно з інформацією на паперовому носії. Але, незважаючи на те, що робота з інформацією стає простішою, виникають інші проблеми, пов'язані з її зберіганням. Основна проблема - це можливість втрати інформації, особливо тієї, що має якусь цінність. До деякого часу цій проблемі не приділяли належної уваги, і найкращим захистом від втрати інформації було обмеження фізичного доступу до її носія.

Однак стрімкий розвиток технологій і зростання залежності бізнесу від ІТ змушують компанії уважніше та вдумливіше підходити до питань інформаційної безпеки, адже що складніша інформаційна система, то більше існує можливих ризиків. Особливо це актуально для компаній, які ведуть будь-яку комерційну діяльність онлайн або надають сервіси в Інтернеті.

Найчастіше дані, що зберігаються в компанії, підлягають захисту відповідно до законів, або становлять комерційну таємницю. Тому підприємства зацікавлені у використанні захищених сховищ інформації. На ринку існує пропозиція корпоративних хмарних сховищ, проте представлені вони іноземними розробниками. Деякі з таких рішень, на відміну від персональних хмарних сховищ, надають у користування окремий сервер або кілька серверів, розташованих у дата-центрах компанії, що надає послугу. Однак, виникає проблема з можливістю обробляти та зберігати персональні дані в такому сховищі у зв'язку зі змінами в законодавстві. Оскільки використання для хмарних сховищ серверів, розташованих за кордоном, є небажаним, виникає потреба у створенні власних сховищ інформації.

Корпоративні рішення в галузі хмарних сховищ даних мають низку переваг перед зберіганням даних на сервері в локальній мережі підприємства, що зараз

використовується в більшості випадків. Важливим аспектом, що використовується в хмарних сховищах, є поділ користувачів за рівнем доступу до даних. Також відкривається можливість роботи користувача зі своїми даними з будь-якого комп'ютера підприємства.

Метою цієї роботи є розробка захищеного хмарного сховища підприємства, що має власну систему аутентифікації, використовує різні способи авторизації, дає змогу зберігати різноманітну інформацію (файли та бази даних).

Для реалізації поставленої мети необхідне виконання таких завдань:

- проєктування хмарного сховища підприємства;
- розподіл ролей користувачів;
- визначення методів захисту сховища підприємства;
- вибір алгоритму шифрування інформації;
- розробити програму для доступу клієнтських пристроїв у сховище.

1 АНАЛІЗ ТЕХНОЛОГІЙ ЗБЕРІГАННЯ ІНФОРМАЦІЇ

1.1 Існуючі підходи до зберігання інформації

З розвитком цифрової техніки кількість інформації значно збільшується. Зараз практично кожна подія в нашому житті знаходить відображення в цифровому варіанті. У результаті цього людина не може вже обробляти, зберігати і передавати такі обсяги інформації без допомоги автоматизованих інформаційних систем.

Автоматизована інформаційна система - комплекс інформаційних, програмних, технічних, організаційно-методичних та інших необхідних засобів, що забезпечують збирання, оброблення, зберігання, передавання даних, а також маніпулювання ними для розв'язання різних завдань. Далі в роботі під інформаційною системою розумітимемо автоматизовану інформаційну систему.

Функціонування інформаційної системи забезпечує певне сховище даних, у якому міститься вся оброблювана і надана цією системою інформація.

Сховище даних - це сукупність програмно-апаратних засобів, що забезпечує зберігання даних, а також їх коректне надання керівній системі. У разі необхідності забезпечення безперервної роботи інформаційної системи або високої цінності збережених даних може використовуватися резервне копіювання.

Можна виділити кілька основних типів інформаційних систем залежно від призначення:

- корпоративні;
- персональні;
- електронні системи інформування.

Як корпоративну інформаційну систему розглядають інформаційну систему масштабу підприємства. Головним завданням такої системи є інформаційна підтримка виробничих, адміністративних і управлінських процесів

Існує велика кількість способів організації зберігання даних. Сховище даних для інформаційної системи кожного типу матиме деякі особливості реалізації.

Найпростіший і найпоширеніший варіант - це локальне зберігання інформації на персональному комп'ютері, використовуючи місце на жорсткому диску. Очевидним недоліком такого способу зберігання інформації є те, що працювати з даними не можна можна тільки з цього персонального комп'ютера. При зберіганні інформації на зовнішньому носії доступ до інформації вже не залежить від можливості роботи на конкретному комп'ютері, але значно збільшується можливість втрати інформації внаслідок технічної несправності носія або його втрати. Останнє також може призвести до розголошення інформації, що досить часто є неприпустимим.

Для зручнішого способу зберігання інформації та забезпечення доступу до неї використовують локальні мережі. Наявність спільного доступу до ресурсу забезпечує можливість його спільного використання з будь-якого персонального комп'ютера, що знаходиться в мережі. Таким мережевим ресурсом може бути папка, у якій зберігається інформація. Однак комп'ютер, на якому розміщена така папка, увесь час має бути доступним, тобто увімкненим і мати постійне підключення до локальної мережі.

Тому, якщо існує необхідність постійного доступу до даних кількох користувачів, використовують сервери для зберігання інформації. Сервери дають змогу зберігати досить великі обсяги інформації, на відміну від персональних комп'ютерів або переносних носіїв. Тому створення сховищ даних з використанням серверів досить часто застосовується на підприємствах. Сховище даних - це сукупність програмно-апаратних засобів, що забезпечує зберігання даних, а також їх коректне надання.

1.2 Технології для створення систем зберігання даних

Існує кілька основних технологій для створення систем зберігання даних. Для окремого персонального комп'ютера під системою зберігання даних можна

розуміти внутрішній жорсткий диск або систему з внутрішніх і зовнішніх жорстких дисків. Якщо ж мова заходить про корпоративну систему зберігання даних, то традиційно можна виділити кілька технологій організації зберігання даних: DAS, JBOD, NAS і SAN.

DAS (Direct-attached storage) - сховище з прямим підключенням. DAS належить до системи зберігання комп'ютера, тому що безпосередньо підключається до сервера або персонального комп'ютера, без використання мережі. (Рис. 1.1) Хоча DAS можна використовувати як жорсткий диск персонального комп'ютера, здебільшого його використовують як виділений об'єм для зберігання даних, безпосередньо під'єднаний до сервера. SATA, eSATA, SAS є основними інтерфейсами і протоколами, які використовуються технологією DAS[1].

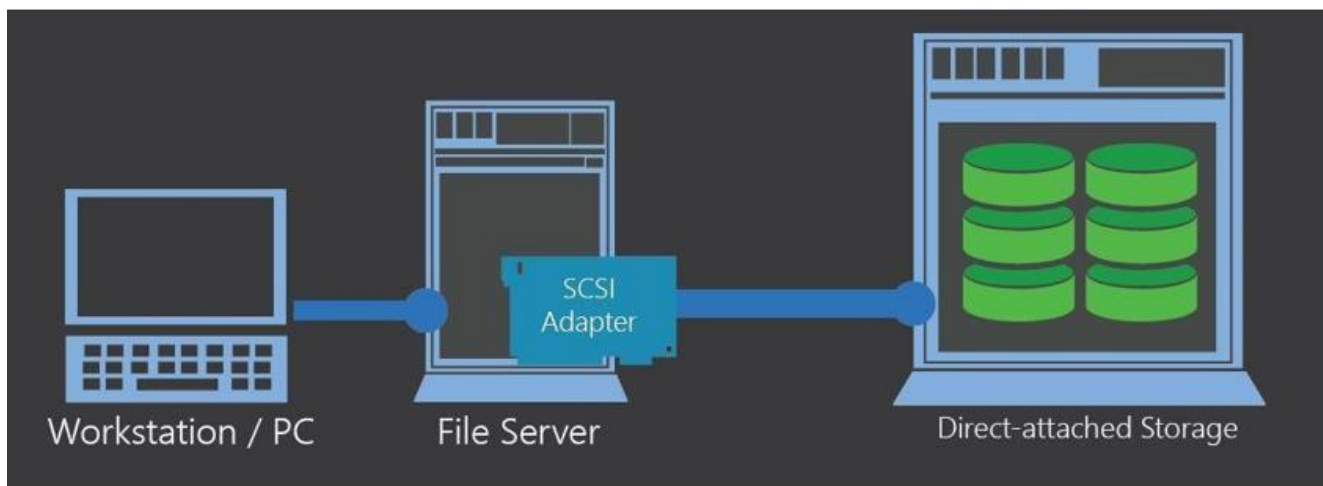


Рисунок 1.1 - Технологія зберігання DAS

JBOD ("just-a-bunch-of-disks") - дослівно "просто зв'язка дисків". (рис. 1.2) Це означає, що диски надано для комп'ютера так, начебто безпосередньо під'єднано до нього. Ця технологія дає змогу збільшувати ємність DAS або сервера. Диски підключаються до одного або декількох серверів через SAS або eSATA [1].

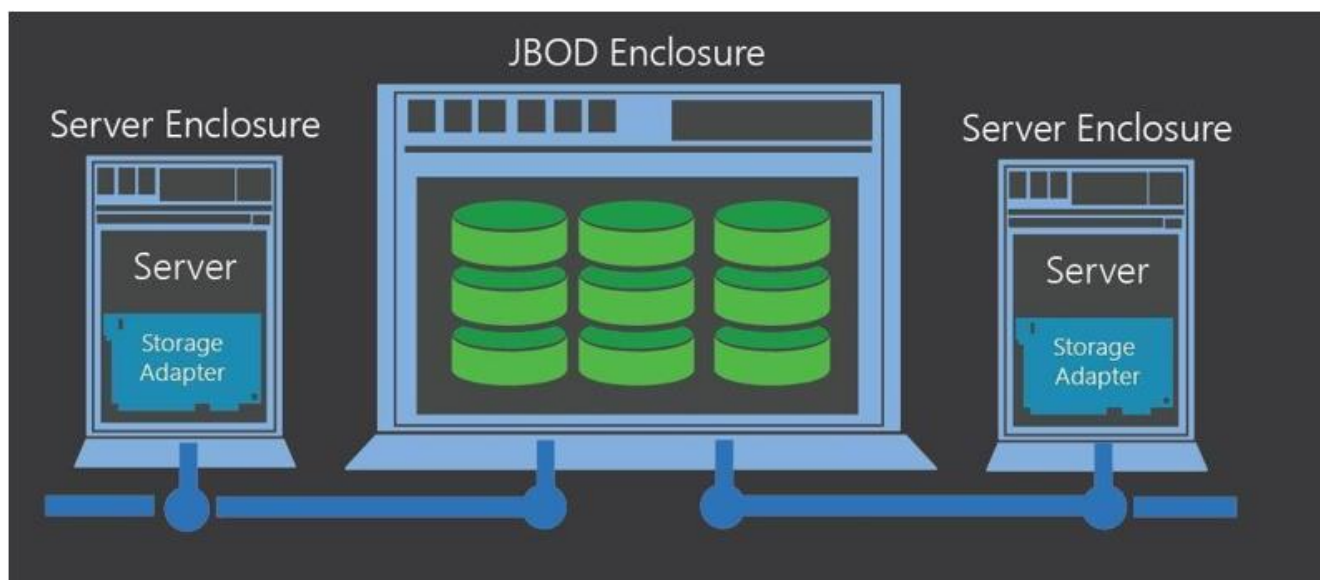


Рисунок 1.2 - Технологія зберігання JBOD

NAS (Network-attached storage) - мережеве сховище. (Рис. 1.3) NAS під'єднується до мережі та стає доступним у вигляді мережевих ресурсів. На відміну від технологій DAS, не використовує пряме підключення до комп'ютера. NAS підтримує Ethernet з'єднання, що дає змогу керувати дисковим простором [1].

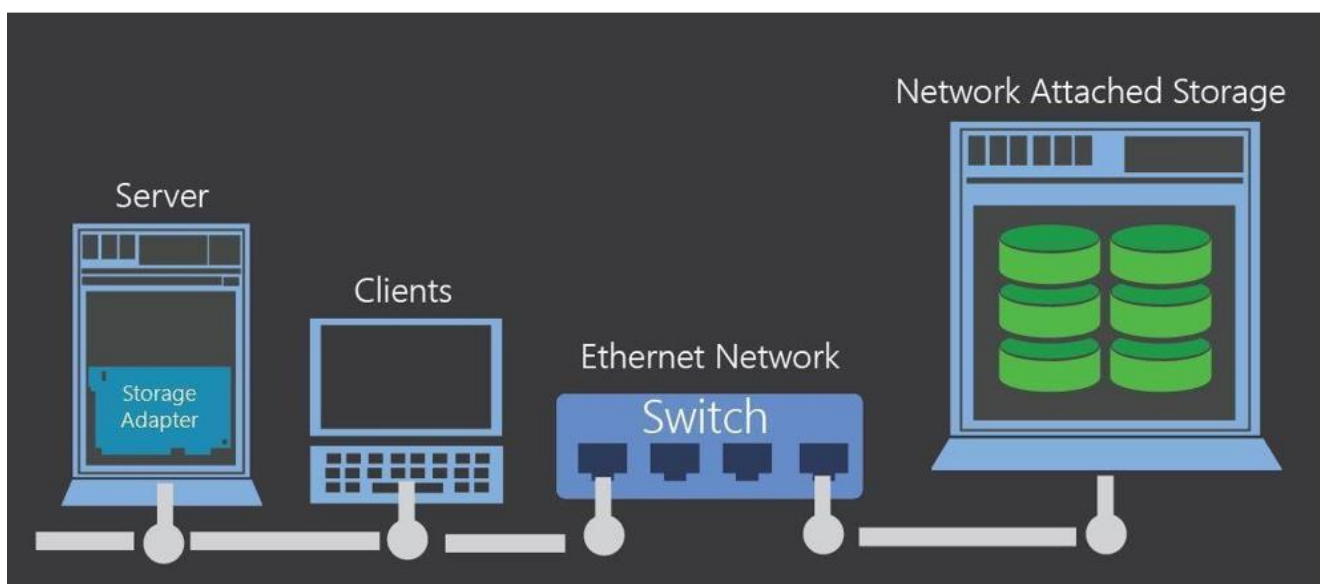


Рисунок 1.3 - Технологія зберігання JBOD

SAN (Storage area network) - мережа зберігання даних. (рис. 1.4) SAN складається із запам'ятовуючого пристрою, з'єднувальної мережевої інфраструктури (комутаторів тощо) і серверів, під'єднаних до цієї мережі. Технологія SAN забезпечує безперервний і швидкий доступ до великих обсягів даних. Для SAN використовують окрему мережу, зазвичай, недоступну іншим пристроям у локальній мережі. SAN можна керувати через протокол SNMP або пропрієтарний протокол управління (proprietary management protocol) [1].

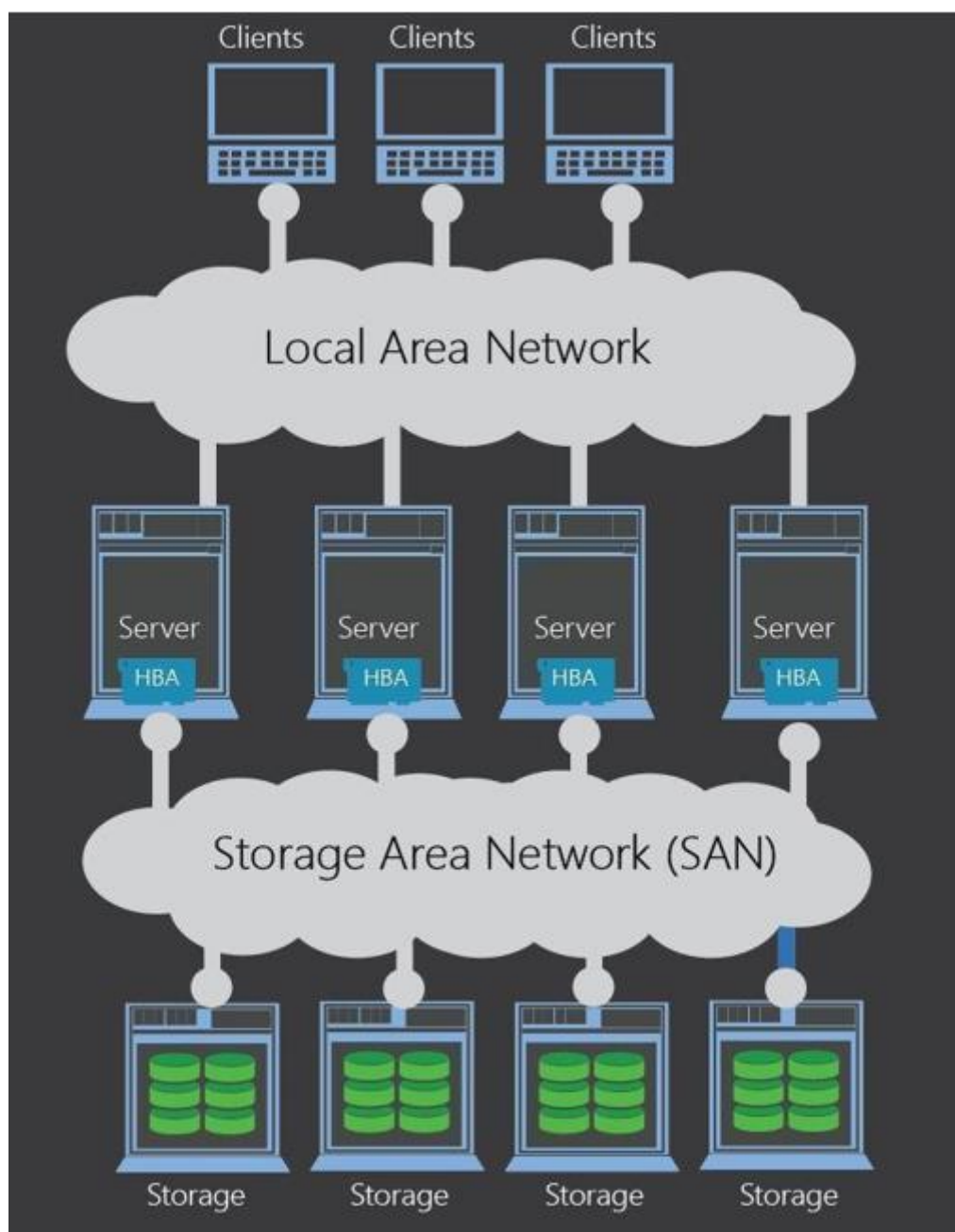


Рисунок 1.4 - Технологія зберігання JBOD

Залежно від вимог до системи для зберігання інформації вибирається технологія її побудови. За необхідності можливе створення системи, що використовує кілька технологій зберігання даних.

У міру розвитку інформаційної системи компанії виникає необхідність збільшення ресурсів зберігання, тому вимоги до систем, які контролюють мережі та системи зберігання й обробки даних, високі [2].

Масштаби основних виробничих процесів, розміри підприємства, використовувані платформи та програмне забезпечення, а також сучасне законодавство формують критерії вибору як самих систем зберігання, так і способів підключення або методів доставки інформації кінцевим споживачам.

Дані та їхня структуризація також визначають тип сховища інформації. Якщо дані не потребують структуризації, то можливе створення файлового сховища. Робота зі сховищем файлів здійснюється через клієнтський додаток, який надає доступ до файлів. Головною метою такої системи є надання можливості зручного способу управління корпоративною інформацією та створення єдиного простору для зберігання будь-яких файлів[3].

1.3 Зберігання інформації з використанням баз даних речей

Бази даних розглядають як надійне сховище структурованих даних, забезпечене спеціальним механізмом для їх ефективного використання в інтересах користувачів (процесів). Таким механізмом є система управління базою даних. Під системою управління базою даних розуміють програмні або програмно-апаратні засоби, що реалізують функції управління даними, такі як перегляд, сортування, вибірка, модифікація, виконання операцій визначення статистичних характеристик тощо. Бази даних можуть розміщуватися на

комп'ютері користувача (персональна база даних), але частіше використовується спеціальний сервер[4].

Для занесення даних у сховище потрібна їх попередня структуризація. Дані зберігаються в системі таблиць, пов'язаних за ключовими полями. Вибір даних зі сховища здійснюється шляхом формування складних запитів, які звертаються до кількох таблиць. Такий спосіб роботи з даними, з обходом багатьох таблиць, є досить ресурсномістким. Прагнення зниження часу обробки інформації в базах даних призвело до появи розподілених баз даних. Розподілені бази даних можна розглядати як підвид розподілених обчислювальних систем, що займаються обробкою даних. Розподілена обчислювальна система складається із сукупності оброблювальних елементів (не обов'язково однорідних), що з'єднані між собою за допомогою комунікаційної мережі та взаємодіють між собою під час розв'язання певної спільної задачі. Можна виділити дві переваги такої системи:

- збільшення потужності системи під час розв'язання спільного завдання;
- можливість автономної роботи окремих елементів системи.

Таким чином, можна визначити розподілену базу даних як сукупність логічно взаємопов'язаних баз даних, розподілених у комп'ютерній мережі, та розподілену СУБД як сукупність програм, призначених для управління розподіленою базою даних. У разі виходу з ладу окремих частин така система продовжить працювати, однак у користувача не буде можливості працювати з інформацією, яка знаходиться в частині бази даних, що вийшла з ладу. Також застосування розподілених баз даних ускладнює контроль над цілісністю даних та їхній захист.

Усі згадані вище технології та способи зберігання інформації мають певні недоліки. Серед недоліків можна виділити локальність розміщення інформації та неможливість забезпечення постійного доступу, великий час простою через регламентні роботи, такі як роботи на сервері або в мережі. Хмарні технології допомагають у вирішенні цих проблем. Тому використання звичайних локальних або розподілених серверних технологій не відповідає всім вимогам користувачів,

тому що в сучасному світі користувачеві важлива постійна можливість швидкого і зручного доступу до інформації, а ці технології не завжди можуть її забезпечити.

1.4 Хмарні технології

На сьогоднішній день, кожна людина, яка активно використовує персональний комп'ютер, так чи інакше стикалася з поняттям хмарних технологій. З'явившись близько 10 років тому, перші хмарні сховища істотно змінили уявлення користувачів про зберігання інформації. Хмарний сервіс для користувача являє собою певний простір на віддаленому сервері, в якому він може працювати зі своїми файлами, завантажуючи і видаляючи їх, а також надаючи іншим користувачам у спільний доступ [5].

Незважаючи на те, що хмарні сховища з'явилися відносно недавно, вони широко поширені серед користувачів, які працюють з інформацією, до якої часто потрібен доступ. Відмінність від попередніх способів зберігання інформації в тому, що користувач може працювати з інформацією з будь-якого пристрою (персональний комп'ютер, планшет тощо), що має доступ до інтернету.

Для користувача використання "хмари" досить зручне, тому що йому виділяється певний обсяг пам'яті, представлений однією папкою. Користувач працює в "хмарі", при цьому він не знає, де саме зберігається його інформація. Для користувача "хмара" - це його папка. З вищевказаного випливає, що створення хмарних сховищ - це найперспективніша технологія зберігання інформації.

Система зберігання даних у "хмарі" використовує розподілене зберігання даних на серверах. Звідси виникають загрози безпеці інформації, що зберігається, але, зазвичай, інформація користувачів не має особливої цінності, на відміну від корпоративної інформації.

Однак, зручність хмарних сервісів призводить до використання публічних хмарних сховищ користувачами у своїй роботі, тому що користувачам простіше вирішувати завдання знайомим, хоч і побутовим методом. Вони не замислюються про збереження інформації, забезпечення її безпеки та збереження. Такий стан справ, у кращому разі викликає суперечки між різними службами компанії, а в гіршому - призводить до порушення системи безпеки. Крім загрози безпеці, використання працівниками публічних хмарних сервісів призводить до проблем, пов'язаних із законодавством. Простими заборонами виправити таку ситуацію досить складно. Компромісом може стати використання корпоративних хмарних сервісів як елемента інформаційної інфраструктури компанії[6].

Завдання зберігання інформації на підприємстві, звичайно, не обмежується тільки можливістю колективного доступу до файлів. Крім цього, сховище інформації має володіти достатнім обсягом, щоб справлятися з постійно зростаючою кількістю інформації.

У відповідь на зростаючу потребу з боку компаній на ринку поступово з'являються різні пропозиції в галузі хмарних технологій для підприємств.

Головною перевагою хмарного зберігання даних для компанії, є готова інфраструктура хмари. Це дає змогу компанії економити на трудовитратах, пов'язаних із розробкою власних сервісів.

Світовий ринок хмарних рішень і послуг зростає настільки інтенсивно, що передбачити темп його збільшення виявляється на практиці досить важко, тому дані провідних аналітичних компаній деколи сильно відрізняються. Проте всі вони фіксують одні й ті самі тенденції: швидкий темп зростання витрат на cloud computing (хмарні обчислення), а також супутнього ринку сервісів, центрів обробки даних і трафіку даних у таких системах.

У квітні 2017 року аналітична компанія 451 Research опублікувала результати дослідження, що показали нову цінову війну, яку розв'язали лідери ринку хмарних обчислень. У повідомленні від 20 квітня 2017 року йдеться про те, що, якщо раніше цінова конкуренція розвивалася тільки в сегменті віртуальних машин у хмарі, то тепер вона торкнулася сервісів зберігання даних[7].

Хмарні сервіси, представлені на ринку, можна розділити на кілька основних видів залежно від послуг, що надаються.

РaaS (платформа як послуга) - модель надання хмарних обчислень, за якої споживач отримує доступ до використання інформаційно-технологічних платформ: операційних систем, систем управління базами даних, зв'язуючого програмного забезпечення, засобів розроблення та тестування, розміщених у хмарного провайдера. У цій моделі вся інформаційно-технологічна інфраструктура, включно з обчислювальними мережами, серверами, системами зберігання, цілком управляється провайдером. Провайдер так само визначає набір доступних для споживачів видів платформ і їхніх керованих параметрів. Споживач може використовувати платформи на свій розсуд, створюючи їхні віртуальні екземпляри, встановлюючи, розробляючи і тестуючи на них прикладне програмне забезпечення. При цьому існує можливість динамічної зміни кількості споживаних обчислювальних ресурсів [8].

Найчастіше РaaS використовується програмістами, які спільно працюють над різними проєктами. У цьому випадку всі або частина розробників отримують доступ до єдиного середовища розробки віддалено. Відповідно, всі вони потребують достатньої кількості системних ресурсів, а також інструментів спільної роботи.

DBaaS (Database as a Service, база даних як послуга) - це різновид РaaS. Використовуючи DBaaS, користувач може отримати доступ до бази даних будь-якого типу за запитом. Користувач може швидко розгорнути БД на будь-якому класі обладнання в середовищі обраної ним програмної платформи (операційної системи). Компанія IBM, наприклад, надає доступ до масштабованої та повністю керованої бази даних через об'єктно-орієнтовані API.

SaaS (програмне забезпечення як послуга) - одна з форм хмарних обчислень, модель обслуговування, за якої передплатникам надається готове прикладне програмне забезпечення, яке повністю обслуговується провайдером. Постачальник у цій моделі самостійно управляє додатком, надаючи замовникам

доступ до функцій із клієнтських пристроїв, як правило, через мобільний застосунок або веб-браузер[9].

Основна перевага моделі SaaS для споживача послуги полягає у відсутності витрат, пов'язаних зі встановленням, оновленням і підтримкою працездатності обладнання та програмного забезпечення, що працює на ньому.

Прикладом SaaS може слугувати MicrosoftOffice 365. Корпорація Microsoft надає за моделлю SaaS доступ клієнтам до MS OfficeSuite (OfficeWebApps) поряд із SharePointServer, ExchangeServer та іншими сервісами і додатками.

IaaS (інфраструктура як послуга) надається як можливість використання хмарної інфраструктури для самостійного управління ресурсами оброблення, зберігання, мережами та іншими фундаментальними обчислювальними ресурсами, наприклад, споживач може встановлювати й запускати довільне програмне забезпечення, що може охоплювати операційні системи, платформне та прикладне програмне забезпечення. Споживач може контролювати операційні системи, віртуальні системи зберігання даних і встановлені додатки, а також мати обмежений контроль над набором доступних мережевих сервісів (наприклад, міжмережовим екраном, DNS). Контроль і управління основною фізичною і віртуальною інфраструктурою хмари, зокрема мережею, серверами, типами операційних систем, що використовуються, системами зберігання, здійснюється хмарним провайдером[10].

У цьому разі постачальник послуги надає в оренду обчислювальні ресурси. Це може бути сукупність віртуальних машин, сховищ даних, мережевих елементів різних типів. За допомогою IaaS користувач отримує можливість швидко розгортати копії ОС, запускаючи віртуальні копії низки програмних пакетів. У цьому разі немає необхідності розгортати власну мережеву інфраструктуру. Все необхідне надається постачальником IaaS. При цьому таке середовище практично завжди є гнучким і масштабованим.

Нещодавно з'явився новий сервіс із захисту хмарних сховищ для бізнесу - Securityas a Service.

Securityas a Service (SecaaS) - це бізнес-модель, у якій сервіс-провайдер послуг безпеки інтегрує свої сервіси в корпоративну інфраструктуру замовника. Основні напрямки даної послуги це - антивірусна та антиспам-фільтрація електронної пошти в хмарі, сервіси із захисту від DDoS-атак [11].

Однією з провідних компаній у цій галузі є Incapsula, яка запустила першу точку присутності сервісу захисту в одному з провідних ЦОД.

Для клієнтів, які використовують сервіси захисту веб-сайтів, відкриття точки присутності Incapsula дає змогу:

- зменшити час відгуку і time-to-first-byte;
- кешувати трафік на серверах, що робить ще ефективнішим вбудований CDN від Incapsula;
- обробляти трафік у дата-центрі.

Цей сервіс не зовсім зручний для внутрішнього сховища підприємства, оскільки здебільшого використовується для захисту веб-ресурсів.

Незважаючи на активний розвиток хмарних технологій у світі, для нашої країни "хмари" як корпоративний сервіс досі є чимось новим. Поширення широкосмугових мереж доступу в Інтернет вирішило основні проблеми використання хмарних технологій. Нині практично кожен користувач має високошвидкісне підключення до Інтернету, причому постійно і з будь-якого пристрою. І, якщо користувачі активно використовують публічні хмарні сховища в особистих цілях, то організації не достатньо довіряють хмарним технологіям, щоб зберігати з їхньою допомогою корпоративні дані. Звісно, ці побоювання не безпідставні, враховуючи, що дані, які зберігаються в "хмарі", фактично належать компанії-провайдеру, яка володіє сервером.

Однак, з кожним роком все більше організацій розуміють зручність використання хмарних технологій. Про це свідчить той факт, що за оцінкою iKS-Consulting хмарних послуг за підсумками 2015 року зріс майже на 40% порівняно з 2014 роком [12].

1.5 Приклади хмарних рішень зберігання корпоративної інформації

Одним з найбільш відомих хмарних рішень для підприємств є MICROSOFT AZURE.

Платформа Microsoft Azure складається зі служб додатка, що виконуються в центрах обробки даних Microsoft і доступні через Інтернет [13].

Однією з найбільш базових функцій хмарної платформи є виконання додатків. Microsoft Azure передбачає для цього три варіанти: Віртуальні машини, Веб сайти та Хмарні служби. Кожен із цих трьох підходів можна використовувати як окремо, так і комбінувати їх, створюючи застосунок, що спільно використовує дві та більше моделі виконання.

Компонент "Віртуальні машини" платформи Microsoft Azure пропонує можливість створювати віртуальні машини з погодинною оплатою цих віртуальних машин. Цей підхід відомий під назвою "Інфраструктура як послуга" (IaaS).

Один із найпоширеніших способів використання хмарних платформ - це запуск у ньому вебсайтів і веб-додатків. У рамках цієї моделі виконання користувач отримує кероване веб середовище на основі служб Internet Information Services (IIS), з можливістю перенести наявний вебсайт (що працює під управлінням IIS) на платформу Microsoft Azure в незмінному вигляді, або ж створити новий сайт прямо в хмарі. "Веб сайти" платформи Microsoft Azure працюють як у режимі поділу ресурсів (коли ваш вебсайт виконується спільно з іншими сайтами в рамках однієї віртуальної машини), так і в режимі виділення окремої віртуальної машини для сайту.

Компонент "Хмарні служби" платформи Microsoft Azure - це приклад того, що зазвичай називають "Платформа як послуга" (PaaS). Щоб скористатися ним, користувач створює застосунок на основі обраної технології, як-от C#, Java, PHP, Python, Node.js або якоїсь іншої. Потім код виконується у віртуальних машинах

(званих екземплярами) під управлінням версії операційної системи Windows Server.

Кожній із трьох моделей виконання платформи Microsoft Azure відведено особливу роль. Віртуальні машини платформи Microsoft Azure забезпечують обчислювальне середовище загального призначення. Веб сайти платформи Microsoft Azure пропонують недорогий веб-хостинг. А Хмарні служби платформи Microsoft Azure - це найкращий вибір для створення масштабованих надійних застосунків із низькими витратами на адміністрування.

Платформа Microsoft Azure дає змогу створювати хмарні віртуальні приватні мережі, з'єднувати додаток у хмарі з локальними машинами, а також інтелектуально розподіляти запити користувачів між центрами обробки даних.

Свої хмарні рішення пропонують більшість компаній у галузі інформаційних технологій.

Bluemix - це пропозиція в серії хмарних рішень IBM® [14]. Це середовище дає змогу розробникам і організаціям швидко та легко створювати, розгортати й адмініструвати додатки в хмарі. Bluemix являє собою реалізацію архітектури IBM Open Cloud Architecture на основі відкритого ПЗ Cloud Foundry, що працює за принципом "платформа як послуга" (Platform as a Service - PaaS). Bluemix надає послуги корпоративного рівня, які можна легко інтегрувати в хмарні додатки, не заглиблюючись у тонкощі їхнього встановлення та налаштування.

Google AppEngine - служба хостингу сайтів і web-додатків на серверах Google. Додатки, що розгортаються на базі AppEngine, мають бути написані на Python, Java, Go або PHP. Середовище виконання Python містить повну реалізацію можливостей самого Python, більшість функцій стандартної бібліотеки мови, обмежену версію Django тощо.

SpiderOak - програмне забезпечення, розроблене компанією SpiderOak Inc., що призначене для резервного копіювання користувацьких даних і їхнього подальшого зберігання в "хмарі". Також SpiderOak дає змогу користувачам синхронізувати дані між комп'ютерами та ділитися ними з іншими користувачами в Інтернеті.

Наразі SpiderOak підтримує пристрої під управлінням операційних систем Microsoft Windows, Mac OS X, і Linux..

Інформація передається на сервер тільки в зашифрованому вигляді, а пароль не передається ніколи, крім тих випадків, коли користувач працює з веб-інтерфейсом. Також є можливість синхронізації файлів і папок на декількох пристроях одночасно і автоматичного усунення дублювання файлів (дедуплікація даних). SpiderOak використовує багаторівневий підхід до шифрування, використовуючи комбінацію 2048 біт RSA і 256-бітове AES.

Історія зміни файлів ведеться за принципом дельта-кодування, щоб заощадити місце, займане файлами. В історії зміни записується тільки відмінність однієї версії файлу від іншої. Очевидним недоліком цієї системи є відсутність авторизації у веб-інтерфейсі.

Центри обробки даних Carbonite захищені охороною, що працює цілодобово 365 днів на рік, а доступ персоналу до даних контролюється за допомогою біометричних сканерів у точках доступу.

Збереження інформації гарантує 128-бітний стандарт шифрування Blowfish. На клієнтському рівні конфіденційність даних забезпечується за рахунок персональних ключів шифрування. Під час передавання даних використовується стандартна технологія Secure Socket Layer (SSL).

Hightail (раніше YouSendIt) дозволяє клієнтам ділитися папками з опцією обмеження доступу до окремих файлів. Власники облікових записів можуть також додавати дату закінчення терміну на доступ до файлів і використовувати захист паролем, перевірку ідентичності, звіти про відстеження файлів.

Під час надсилання даних використовується 128-бітове SSL-шифрування, а для зберігання - 256-бітове AES-шифрування. Hightail пройшли акредитацію SAS 70 Type II, SSAE, SOC 2 Type 2, TRUSTe, PCI, а також відповідають законам HIPAA і GLBA.

Клієнти LiveDrive можуть скористатися опцією SSL під час входу в систему, щоб переконатися, що всі дії між комп'ютерами і хмарними сховищами виконуються в умовах надійного і захищеного з'єднання. Дані зберігаються на

кількох серверах (у разі витоку буде відкрито лише частину даних), для їхнього шифрування використовується AES 256.

Плюс до всього перерахованого вище, передача даних між пристроями і хмарою LiveDrive також шифрується. Клієнти можуть встановити власні FTP акаунти, але за таких умов безпека інформації під час її передання буде вразливою, оскільки FTP не пропонує 100% охорони або протоколів шифрування.

Як і у випадку з іншими хмарними сховищами, SuperSync дає змогу своїм клієнтам обмежувати доступ до певних файлів під час передачі даних. Їхня панель адміністратора пропонує такі опції, як контроль за доступом і протокол дій усіх користувачів. Унікальність їхніх планів полягає в тому, що в разі витоку інформації ви можете віддалено позбутися її.

Безпека файлів досягається завдяки доступу "тільки читання", TLS шифруванню, 256-бітного AES під час передавання даних і протоколу взаємного ідентифікації для забезпечення захищеного зв'язку через інтернет.

Крім того, що Bitcasa може зробити резервні копії всіх ваших файлів на жорсткий диск, він ще має дуже зручний і зрозумілий інтерфейс, а також гарантує захищеність вашої інформації. Компанія використовує пропрієтарну програму, яка перед пересиланням даних на сервери шифрує їх на рівні блоків.

Усі файли, що зберігаються в хмарі, перед завантаженням шифруються, і після того, як вони потрапляють на сервери Bitcasa, доступ до них, крім власника облікового запису, не може отримати ніхто (навіть співробітники компанії).

1.6 Проблеми захисту сховищ інформації

Забезпечення безпеки інформації, що використовується в інформаційних системах, вимагає створення захищених сховищ даних. Способи організації цих сховищ різняться залежно від самої інформаційної системи. Для корпоративних інформаційних систем часто використовують сховища, що забезпечують захист

файлів шляхом поділу доступу користувачів. Для приватної інформаційної системи використовують хмарні сховища, що забезпечують доступ до інформації з будь-якого носія з виходом у мережу Інтернет, під час проходження ідентифікації та аутентифікації. Як захід щодо захисту конфіденційності інформації застосовують криптографічні методи. Для кращого захисту інформації часто застосовують певну сукупність цих способів. Це дає змогу запобігти розкраданню інформації в разі несанкціонованого доступу до сховища[15].

Обмеження доступу до інформації не може повністю гарантувати її безпеку. Інформаційна безпека ґрунтується на здійсненні заходів щодо забезпечення конфіденційності, цілісності та доступності інформації.

Конфіденційність - властивість інформації, за якої доступ до інформації можуть отримати тільки її законні користувачі.

Цілісність - властивість інформації, що гарантує можливість зміни інформації, що захищається, тільки з боку її законних користувачів.

Доступність - властивість інформації, за якої здійснюється безперешкодний доступ для її законних користувачів.

Загрози інформаційній безпеці - можливі впливи на інформаційну систему, які можуть завдати шкоди її безпеці, тобто конфіденційності, цілісності та доступності інформації.

Тому для підвищення безпеки інформації використовують криптографічні методи. Застосування криптографічних методів дає змогу усунути недоліки забезпечення безпеки, що виникають під час використання сховищ даних.

Технології безпечного зберігання інформації (storage security) активно розвиваються останнім часом і на це є кілька причин. По-перше, пристрої для зберігання інформації постійно зменшуються в розмірах, при цьому відбувається збільшення їхньої ємності. Це означає, що ступінь концентрації інформації зростає і в разі втрати носія втрачається більша кількість даних. По-друге, на сьогодні вже склалося уявлення про те, як слід захищатися від зовнішніх загроз. Звичайно, ці проблеми, як і раніше, існують, але існують відпрацьовані методи і

дії, які допомагають вибудувати захист від загроз і знизити ризик втрати інформації. [16]

Системи безпечного зберігання даних спрямовані насамперед на безпеку даних у разі фізичного доступу зловмисника до носіїв інформації. Ризик виникнення подібної ситуації може здатися досить невеликим, проте існує кілька цілком можливих варіантів:

- розміщення сервера в сторонньому дата-центрі;
- ремонт і утилізація носіїв;
- втрата або крадіжка обладнання чи носіїв.

У першому випадку необмежений фізичний доступ до сервера має компанія, що надає послугу зберігання, яка є третьою особою і не належить до організації, що зберігає свої дані.

У двох випадках, що залишилися, зловмисник отримує доступ до інформації безпосередньо з носія. Під час ремонту носія зловмисник може спробувати відновити інформацію на жорсткому диску. А в разі крадіжки зловмисник уже має доступ до інформації з носія без особливих труднощів. Тому деякі організації не передають жорсткі диски, що вийшли з ладу, до ремонтних центрів, а одразу знищують їх і замінюють на нові.

Найочевиднішим вирішенням усіх зазначених проблем є шифрування. Зрозуміло, ключ шифрування має бути секретним і зберігатися окремо від зашифрованої інформації. Залежно від того, що конкретно потрібно шифрувати, обирається технологія шифрування. Можна шифрувати розділи даних цілком, або окремо шифрувати файли і папки. Останній варіант є більш гнучким, оскільки крім шифрування є можливість використовувати поділ доступу до файлів. Однак, це викликає додаткові складнощі, пов'язані зі створенням, поширенням і зберіганням ключів користувачів.

Під час генерації ключів шифрування найкращим варіантом є використання фізичного генератора випадкових чисел, але на практиці цей спосіб є важко реалізованим.

Для зберігання ключів сучасні системи використовують смарт-картки та USB-пристрої із захищеною PIN-кодом пам'яттю. Для використання ключа, що зберігається на пристрої, користувач має ввести коректний PIN-код, а кілька неправильних спроб поспіль призводять до блокування пристрою.

Також слід зазначити особливість - кворум ключів, що зустрічається в системах досить рідко. Вона дає змогу розбивати ключ шифрування на кілька складових частин, причому для відновлення оригінального ключа необхідні не всі частини. Ця технологія дає змогу знизити ризик розкриття ключа шифрування і забезпечити більший рівень доступності даних.

Основна проблема, що виникає під час використання систем безпечного зберігання - це велика кількість серверів. Навіть невелика організація може мати близько десяти серверів, що містять конфіденційну інформацію. Оскільки різні області системи зберігання слід шифрувати, використовуючи різні ключі, то потрібен особливий спосіб зберігання та управління ключами шифрування. Також періодично слід проводити шифрування розділів, змінюючи старі ключі. Тому постає питання про використання централізованого захищеного сервера для зберігання ключів шифрування, яких може бути кілька десятків, для їхньої автоматичної генерації та зберігання.

1.7 Технології захисту хмарних сховищ

Більшість розвинених ІТ-компаній намагаються запропонувати свої рішення в галузі хмарних технологій. Однак, незважаючи на помітну зручність у використанні, "хмари" погано захищені з точки зору інформаційної безпеки. Для доступу до даних у хмарі користувачеві потрібно ввести тільки логін і пароль, які можуть бути вкрадені зловмисником за допомогою різних технічних засобів або зламані шляхом перебору. І навіть якщо користувач акуратно використовує свій логін і пароль, пароль має хорошу захищеність від злому шляхом перебору, це

зовсім не гарантує збереження даних, що зберігаються в хмарі. Розглянутий випадок лише ускладнює зловмиснику доступ до хмари через обліковий запис користувача, але не виключає можливості доступу до даних.

Дослідження комп'ютерної безпеки за останні 10 років показали, що зловмисникам не потрібно бачити дані користувача, щоб вкрати їх. Ризик таких атак особливо актуальний у хмарі, де користувач не має контролю над додатками, якими його файли ділять простір на сервері. Зловмисник може просто завантажити невеликі програми на кілька хмарних серверів. Ці програми не роблять нічого, крім шпигунства за даними інших користувачів.[17] Для усунення цієї проблеми пропонуються різні рішення. Найбільш очевидний спосіб для підприємства - це використання власних серверів для зберігання інформації, які знаходяться у внутрішній мережі і відповідно не доступні для такого роду атак. Ще однією особливістю використання хмарних сервісів є те, що користувачі, як правило, не мають поняття, де зберігаються їхні файли та наскільки забезпечено безпеку їхнього зберігання. Тому фахівці в галузі інформаційних технологій дедалі частіше пропонують використовувати шифрування файлів, що зберігаються на серверах, причому для шифрування різної інформації одного й того самого користувача застосовувати різні ключі [18]. А так як сьогодні для доступу до хмари найчастіше застосовують веб-інтерфейс, то виникають складнощі в реалізації запропонованого підходу.

Однак, більш звичайний підхід до шифрування, без передання секретних ключів для доступу до хмари, вже реалізовано і використовується деякими компаніями.

Наприклад, захищене хмарне сховище файлів Wuala (рис. 1.5). Цей сервіс може працювати з операційними системами Windows, Linux, OS X, а також з Android та iOS.

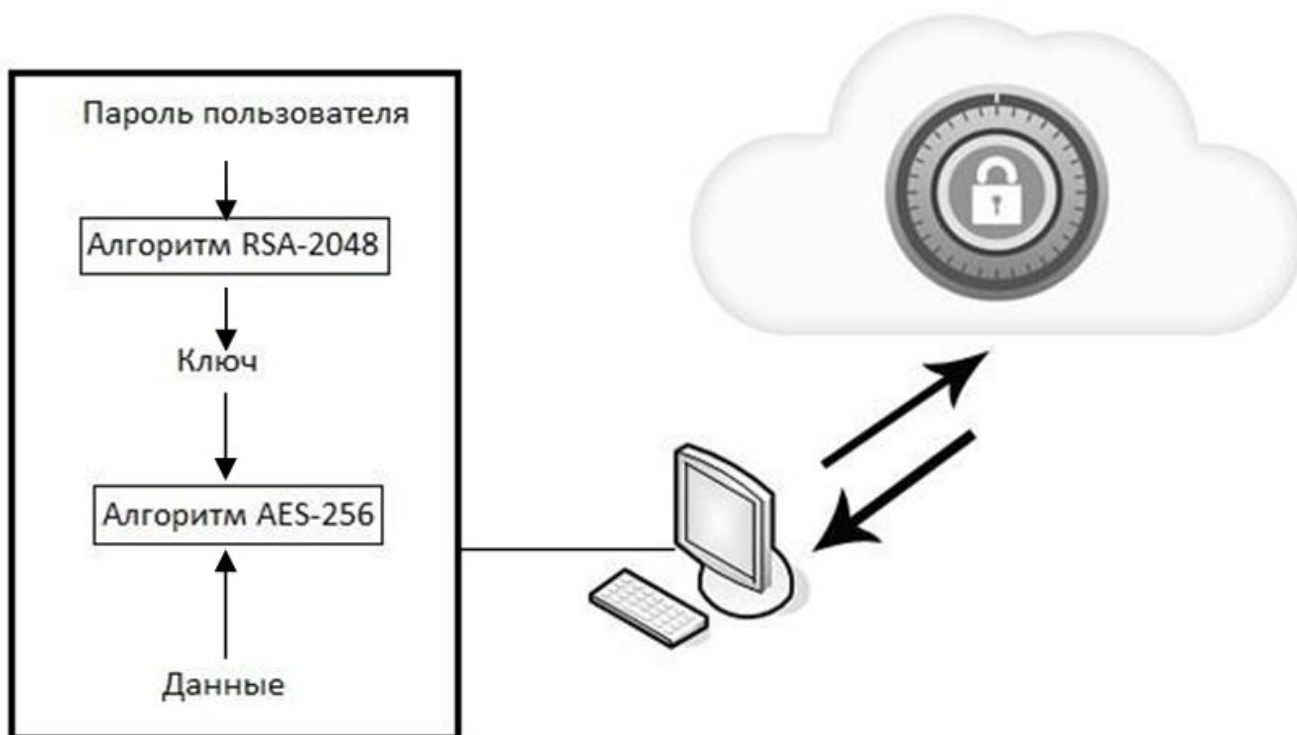


Рисунок 1.5 Шифрування даних для зберігання у Wuala

Систему захисту інформації організовано так: пароль користувача шифрують за алгоритмом RSA-2048, за допомогою отриманого ключа шифрують самі дані користувача (алгоритм AES-256), які в зашифрованому вигляді надсилають на сервер для зберігання [19].

Таким чином, пароль користувача ніде не зберігається у відкритому вигляді і не можлива стандартна процедура відновлення пароля, якщо користувач забуде його.

Зазначена послідовність дій для шифрування інформації впливає на час роботи - всі процеси йдуть помітно повільніше, ніж у звичайних сховищах. Тому цей сервіс підходить тільки для приватного використання.

Корпоративне рішення в галузі хмарних сховищ даних має низку переваг перед зберіганням даних на сервері в локальній мережі підприємства, що зараз використовується в більшості випадків. Однак, серед існуючих підходів немає такого, що задовольняє таким вимогам:

- зберігання інформації на власних ресурсах;
- можливість поділу користувачів за рівнем доступу до даних;

- зберігання різномірної інформації;
- можливість роботи користувача зі своїми даними з будь-якого комп'ютера всередині мережі підприємства.

2 ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Зберігання інформації на підприємствах

Нині зберігання інформації в цифровій формі присутнє на кожному підприємстві незалежно від його розмірів.

Великі підприємства використовують корпоративні інформаційні системи, які обробляють великі обсяги електронної інформації. Для побудови інформаційної системи підприємства потрібна розвинена телекомунікаційна інфраструктура, що об'єднує кабельні, бездротові комп'ютерні мережі центральних офісів з регіональними філіями в єдиний інформаційний простір для передавання різної інформації. Основною метою сховищ інформації в таких системах є створення єдиного простору зберігання даних. У багатьох випадках підприємство використовує власні мережі для передавання даних, тому захист інформації сховища системи ґрунтується на наданні різного рівня доступу співробітникам для роботи з інформацією. Організація самого сховища залежить від структурованості даних.

Більшість таких інформаційних систем будуються за схожою структурою, наведеною на рисунку 2.1

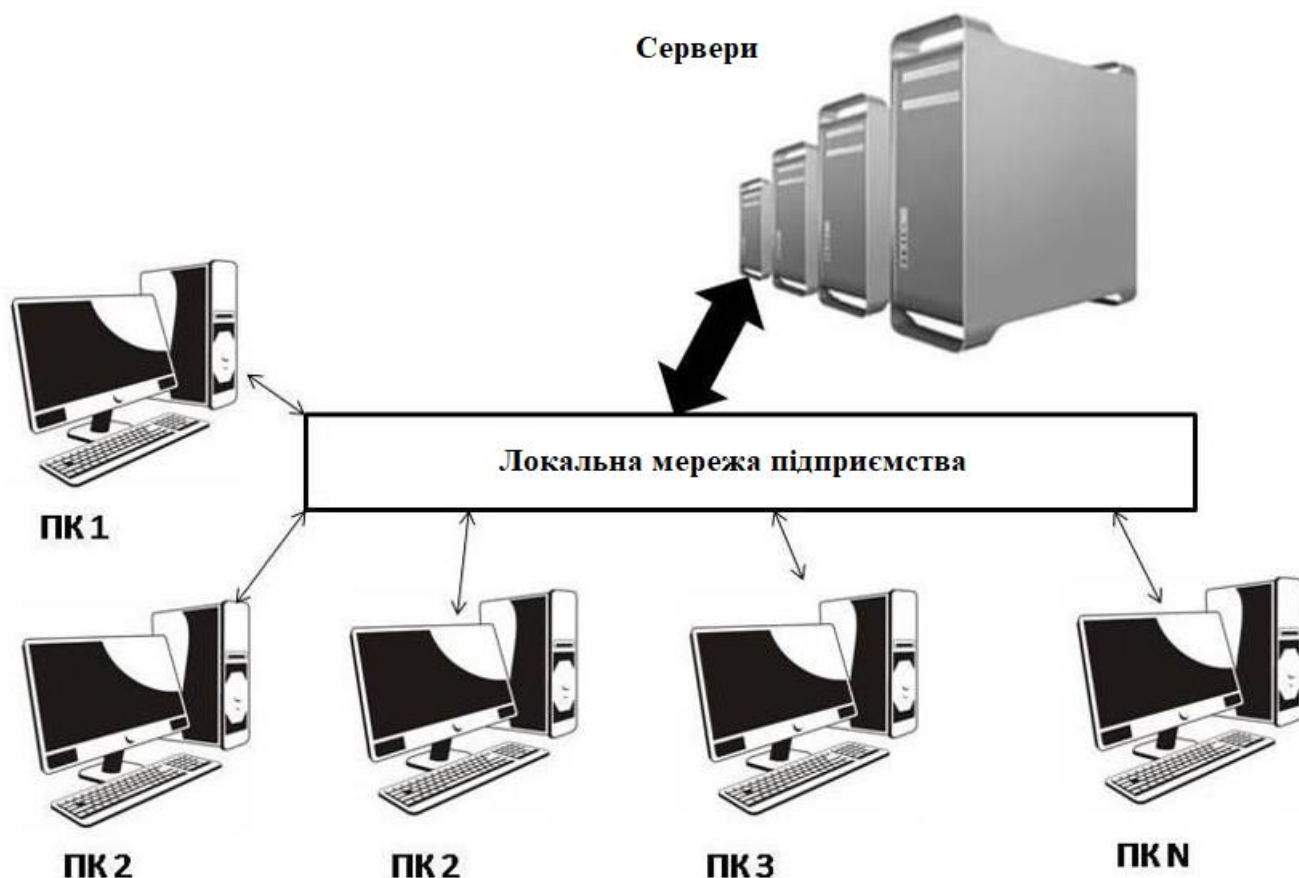


Рисунок 2.1 Структура інформаційної системи підприємства

У мережі підприємства знаходиться один або кілька серверів, які можуть мати різне призначення. Для користувача ці сервери представлені інформаційними ресурсами з локальної мережі. Таким чином здійснюється доступ користувачів до файлів або баз даних, що зберігаються на серверах. Причому, не бажано, щоб уся інформація на сервері зберігалася в єдиному екземплярі. Будь-яка несправність у роботі сервера може спричинити втрату інформації, що зберігається. Частково цю проблему вирішує створення копій, наприклад, образу бази даних, званих "бекапами" (від англ. back up), які зберігаються на окремих серверах. Однак на відновлення роботи сервера і розгортання резервної копії потрібен час і весь цей час користувачі не можуть отримати доступ до інформації на цьому сервері. Тому у великих інформаційних системах застосовують технологію створення "дзеркал". Дзеркальний сервер є повною копією вихідного сервера за змістом, тобто зберігає копії всієї інформації вихідного сервера. Ця технологія дає змогу працювати з інформацією навіть у разі виходу з ладу

основного сервера. Але для такої безперервної роботи потрібна постійна підтримка актуальності інформації, що зберігається на дзеркалі. Щоб підтримувати інформацію на всіх серверах в однаковому стані, використовують синхронізацію. Розрізняють два основні типи алгоритмів синхронізації: односторонній і двосторонній.

Для односторонньої синхронізації характерна наявність основного та додаткового пристрою. Користувач, працюючи з інформацією, вносить зміни на основному сервері, а вже після всі зміни дублюються на додатковий сервер. Зрозуміло, для забезпечення більшої надійності, може бути кілька додаткових серверів, як і "дзеркал". Таке резервне копіювання передбачає періодичне, як правило, щоденне створення копій зміненої інформації. Однак, також можливе примусове копіювання зміненої інформації в разі технічного збою на сервері.

Алгоритми двосторонньої синхронізації рідше застосовуються на підприємствах. Зазвичай користувач змінює інформацію на одному з пристроїв, а потім інформація змінюється на інших пристроях. Тому цей вид синхронізації не дуже підходить для створення резервних копій інформації.

Корпоративні дані є важливою частиною будь-якої організації. Цю інформацію використовують щодня, і якщо її буде поставлено під загрозу або загублено, наслідки негативно позначаться не тільки на репутації організації, а й на поточній діяльності, а також найближчих перспективах розвитку. Резервування даних захищає їх від втрати. Для цього великі організації використовують резервні центри обробки даних. Крім того, організація зобов'язана зберігати не актуальні корпоративні дані ще протягом деякого часу для звітів або з метою дотримання законодавства.

Тож використання компаніями центрів обробки даних значно зросло з появою доступу до широкосмугових інтернет-мереж[20]. Великі компанії задля цього намагаються автоматизувати та стандартизувати всі інформаційні процеси, що призводить до зменшення штату ІТ-фахівців, проте підвищує вимоги до їхньої кваліфікації.

Менш безпечний, але недорогий спосіб зберігання - це резервування в хмару. Однак, поки що він мало поширений через побоювання організацій щодо безпеки такого зберігання та можливості відновлення даних.

Проблеми забезпечення постійного доступу до даних і підтримання актуальності інформації вирішуються використанням хмарних технологій.

2.2 Основні характеристики хмарного сховища підприємства

На відміну від класичної схеми зберігання інформації, при хмарному підході використовується розподілене зберігання інформації, а доступ до інформації можливий з будь-якого пристрою, що знаходиться в мережі. Причому, ресурс для розміщення інформації або її отримання обирається самою хмарною системою, а не користувачем.

По суті, "хмара" для користувача є послугою, що надає зручний доступ до його інформації. Для провайдера, що надає цю послугу, "хмара" - це система з безлічі серверів з резервним копіюванням інформації. (Рис. 2.2)

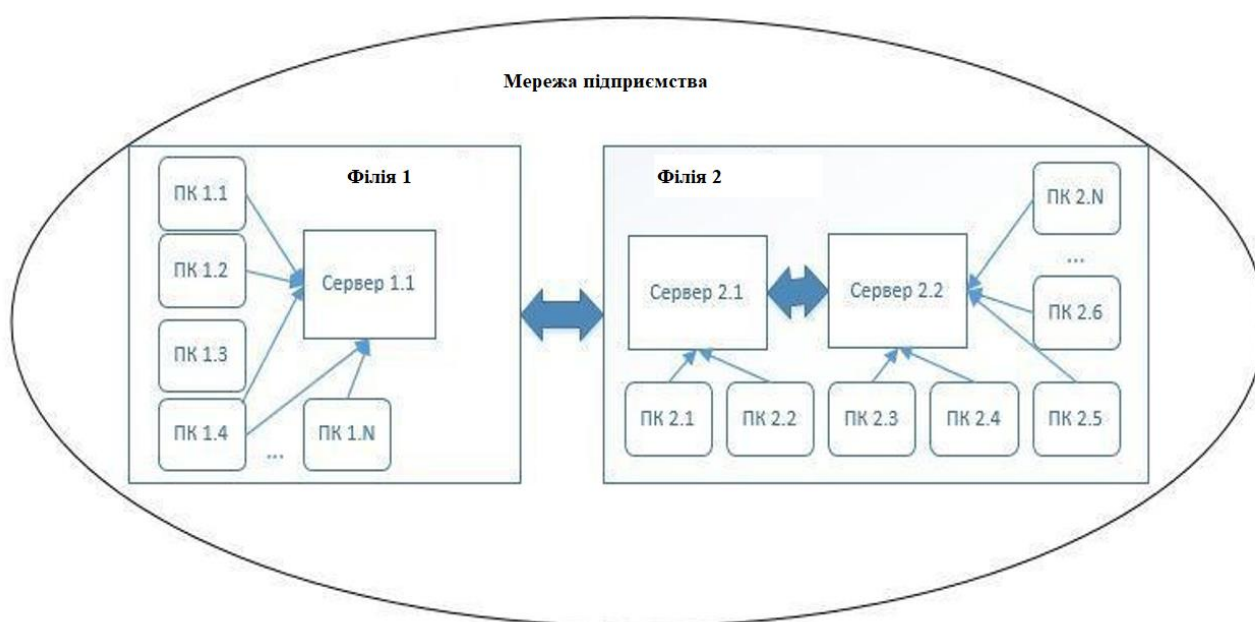


Рисунок 2.2 - Інформаційні ресурси підприємства в корпоративній мережі

Сховище являє собою сукупність інформаційних ресурсів, які можуть розташовуватися в різних філіях підприємства всередині корпоративної мережі. Дані зберігаються у вигляді файлів, які розподілені в ієрархічній структурі каталогів.

Корпоративні мережі називають також мережами масштабу підприємства, що відповідає дослівному перекладу терміна "enterprise-wide networks", який використовується в англomовній літературі для позначення цього типу мереж. Мережі масштабу підприємства об'єднують велику кількість комп'ютерів на всіх територіях окремого підприємства. Вони можуть бути складно пов'язані і здатні покривати місто, регіон або навіть континент. Кількість користувачів і комп'ютерів може вимірюватися тисячами, а кількість серверів - сотнями, відстані між мережами окремих територій бувають такими, що доводиться використовувати глобальні зв'язки.

Зв'язок між різними філіями організовується за допомогою створення VPN (Virtual Private Network - віртуальна приватна мережа). Використання технології VPN під час побудови розподіленої мережі підприємства дає змогу об'єднати локальні обчислювальні мережі всіх територіально рознесених офісів і філій підприємства за допомогою захищеного з'єднання в єдину інформаційну структуру, забезпечує спільне використання корпоративних ресурсів (інформаційних та обчислювальних) і обмін інформацією.

2.3 Клієнтський доступ до сховища

Доступ до сховища користувач отримує через програму "клієнта", використовуючи свій логін і пароль. (Рис. 2.3) Тому отримати доступ до своїх файлів він може з будь-якого комп'ютера всередині цієї мережі

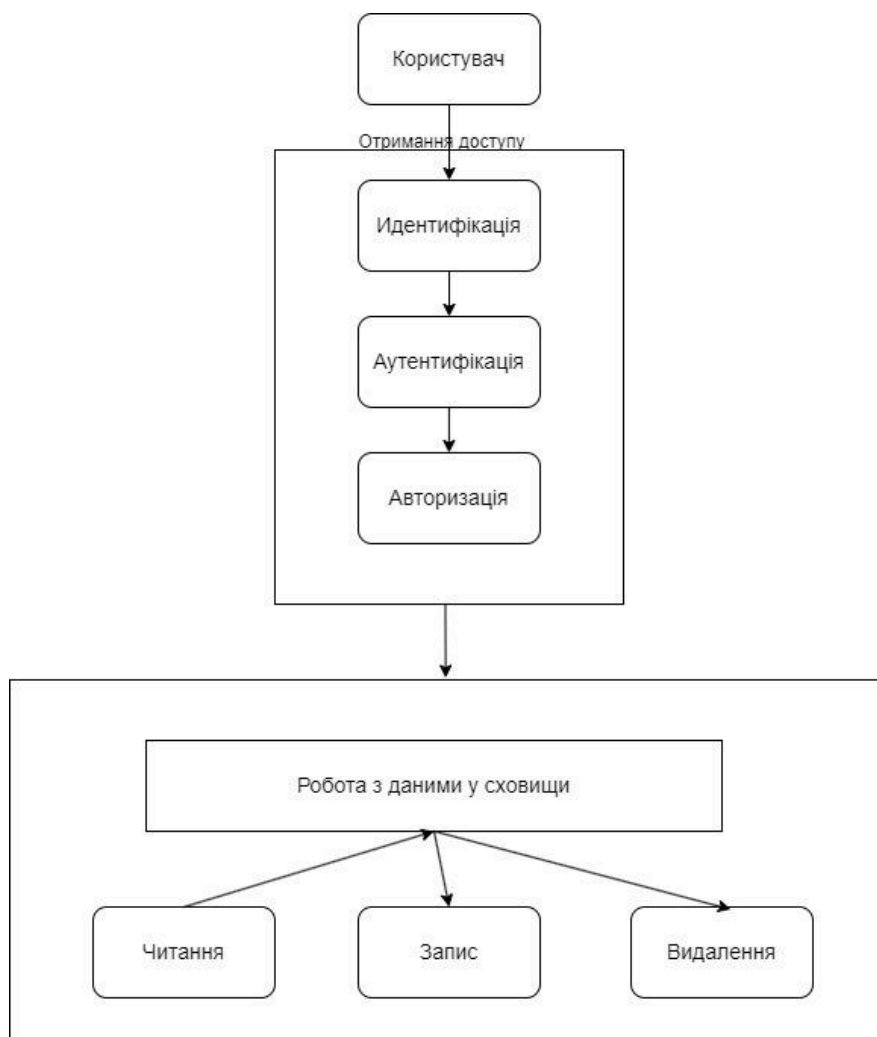


Рисунок 2.3 - Схема доступу до сховища з пристрою-клієнта

Під час увімкнення клієнтського застосунку, користувач проходить процедури ідентифікації та аутентифікації. Процес авторизації визначає права користувача під час роботи з інформацією в сховищі.

Права користувачів можуть помітно відрізнятися і залежать насамперед від посади користувача на підприємстві. Зазвичай усю інформацію, що обробляється і зберігається в інформаційній системі підприємства, можна віднести за призначенням до одного з відділів або кількох відділів. Тому доцільним є створення механізму розмежування доступу користувачів до різних видів інформації. Іноді інформація настільки специфічна, що потрібне забезпечення доступу до неї якомога меншої кількості користувачів. Прикладом такої інформації на підприємстві може слугувати інформація, що становить комерційну таємницю. Однак, природним є те, що начальник відділу має доступ до інформації

всього відділу, а директор може отримати доступ до інформації всього підприємства. Для цих цілей можна використовувати різний рівень доступу до сховища інформації.

Користувачі різних типів розподілені на кілька груп, що мають певний рівень доступу до сховища. (Рис. 2.4)

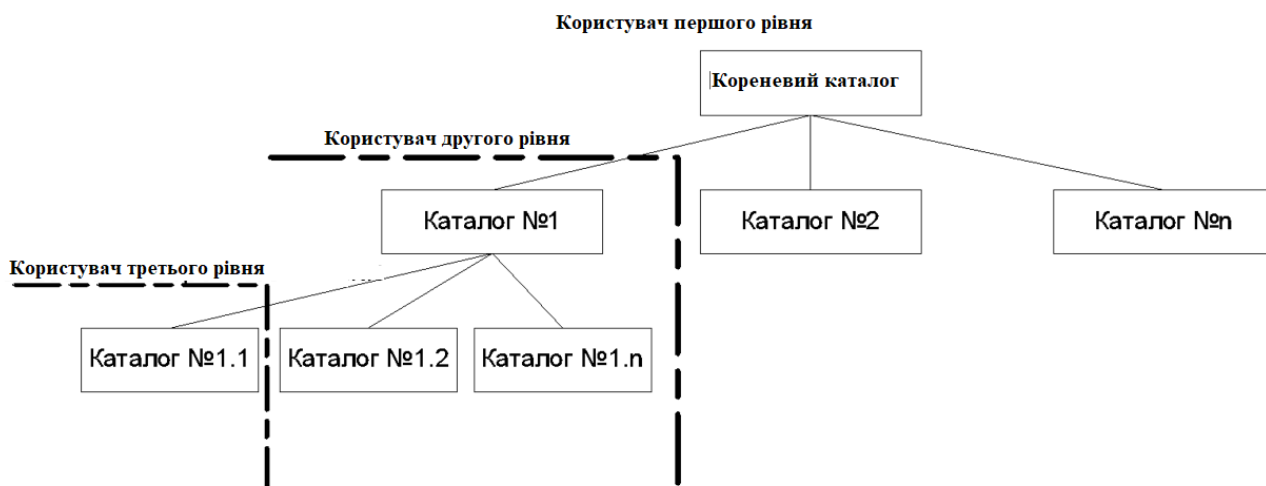


Рисунок 2.4 - Рівні поділу доступу

Користувачі 1го рівня мають повний доступ до всієї інформації у сховищі.

Користувачі 2го рівня мають доступ до інформації, що міститься в каталозі, а також його підкаталогів.

Користувачі 3го рівня мають доступ до інформації, що міститься в каталозі.

Також існує тип користувача Адміністратор, який може створювати групи користувачів, додавати користувачів, вказувати групу користувача, але не має доступу до файлів сховища.

2.4 Сховище для інформаційної системи

Концептуально цю модель зберігання інформації можна застосувати для системи інформування підприємства «Комтел».

Інформаційна система підприємства «Комтел» побудована за архітектурою "клієнт-сервер". (Рис. 2.5) Отже, у системі є пристрої (сервери), що надають певні сервіси, і пристрої (клієнти), які їх використовують. Інформаційна система містить єдиний сервер і кілька клієнтських пристроїв, які розділені на клієнтські термінали та пристрої для адміністрування системи [21].

Ця інформаційна система є розподіленою, тобто пристрої фізично розташовуються віддалено, а передача інформації відбувається за допомогою інформаційно-телекомунікаційних мереж.

Система інформування призначена для зручного і швидкого поширення інформації серед співробітників. Тому основне призначення серверів - це зберігання інформації, що використовується в системі.

У системі є два види клієнтських пристроїв, що відрізняються функціоналом і повноваженнями для роботи.

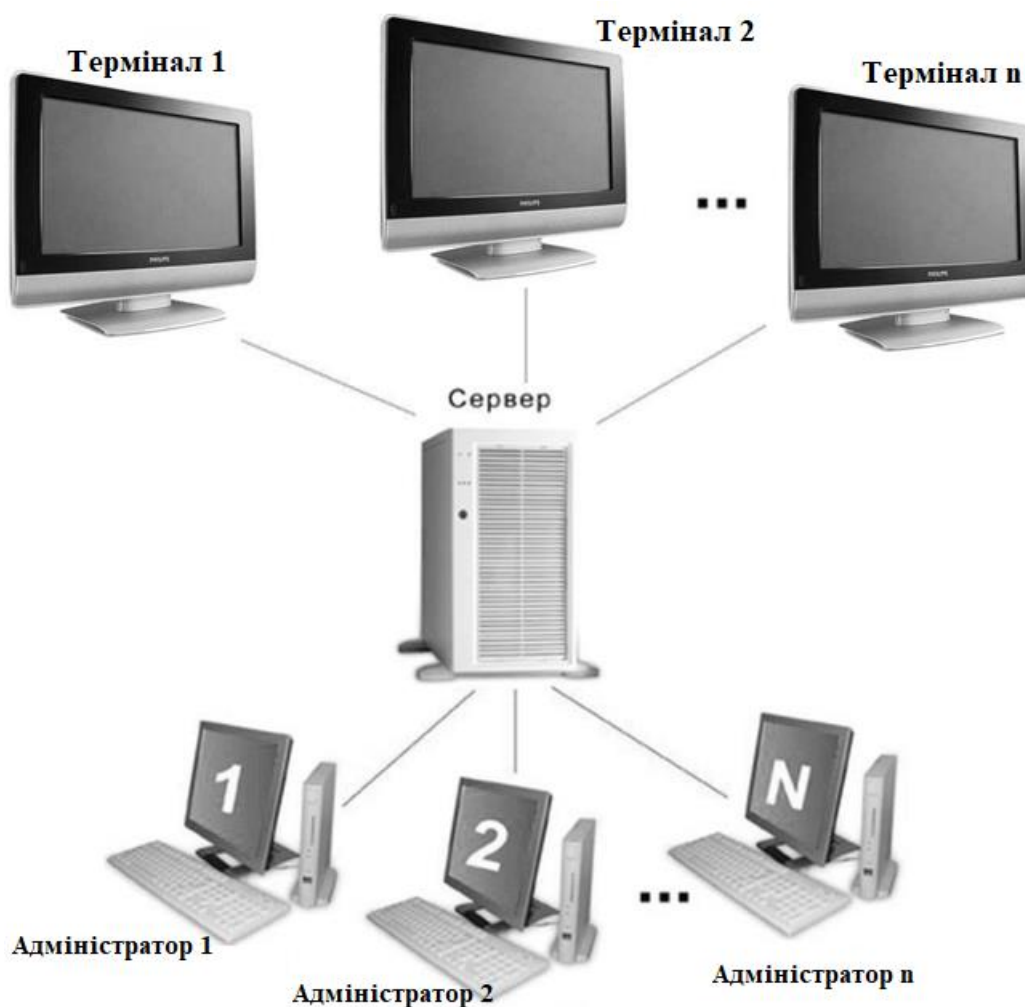


Рисунок 2.5 - Архітектура "клієнт-сервер"

У базі даних міститься інформація про наявних користувачів інформаційної системи, їхні права доступу до сховища файлів. Адміністратор не має доступу до сховища даних. (Рис. 2.6) Його завдання полягає в управлінні базою даних. Адміністратор додає і видаляє користувачів інформаційної системи, для кожного користувача визначає його права доступу.

Файли додаються до сховища, яке має структуру вкладених директорій. Для кожного терміналу є своя директорія, яка містить піддиректорії з шаблонами. Таким чином, у директорії терміналу зберігається вся інформація, що виводиться через нього.

Клієнтські пристрої, що мають доступ до файлового сховища - це модератори та клієнтські термінали.

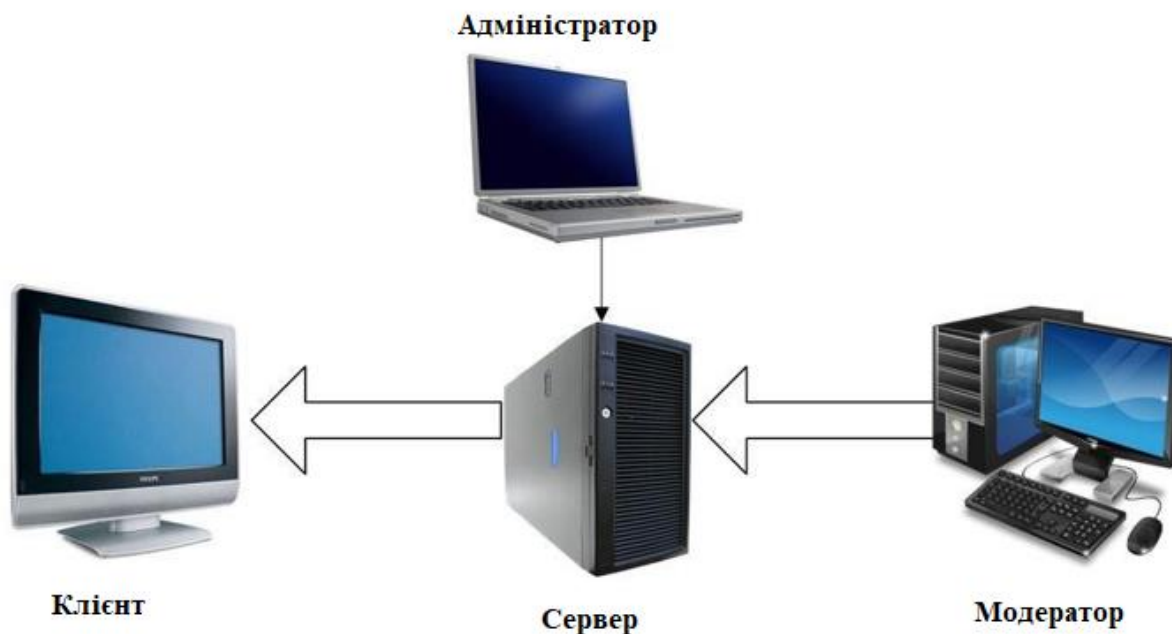


Рисунок 2.6 - Інформаційні потоки системи

Модератори - це користувачі, які додають інформацію в систему, створюючи файли xml за певними шаблонами. Для фізичного доступу модератора в інформаційну систему використовується будь-який ПК.

Клієнтські термінали використовуються для надання інформації кінцевим користувачам, із загального файлового сховища. Як термінали в системі використовуються мікрокомп'ютери Raspberry Pi. Такі комп'ютери розташовані на одній платі. Для Raspberry Pi було розроблено кілька варіантів операційних систем, найпопулярнішою з яких стала Raspbian OS. Raspbian OS була розроблена спеціально для Raspberry Pi і є однією з модифікацій Debian - одного з найпоширеніших дистрибутивів Linux [22]. Для виведення інформації на дисплей використовуються LCD телевізори.

Модератор може тільки додавати файли і видаляти. Причому модератор може видалити тільки файли, додані ним самим. Через застосунок модератор формує файл за шаблоном і надсилає його у сховище [23]. Визначення директорії для запису файлів у сховище відбувається за шаблоном файлу. При отриманні права доступу на запис у сховище файли записуються у сховище за визначеною раніше директорією. Клієнтський термінал має можливість тільки отримувати

файли зі сховища відповідно до обраного шаблону файлів. Термінал формує запит на отримання файлів і відсилає його сховищу. У базі даних перевіряється наявність у клієнтського терміналу прав на доступ до файлів. У разі задовільного запиту сховище пересилає шукану директорію, що містить файли. Оскільки кількість файлів може бути великою, після завершення перетворень усі файли, що відправляються, збираються в архів. Потім цей архів надсилається клієнту, який його розпаковує і записує файли за потрібною директорією[24].

Кількість комп'ютерів-клієнтів у такій системі обмежується кількістю інформаційних панелей, розміщених у корпусах підприємства «Комтел». Передбачається, що така інформаційна панель має бути в кожному корпусі на кожному поверсі підприємства «Комтел». Тому приблизна кількість пристроїв, що отримують інформацію зі сховища, близько 40.

Кількість користувачів-модераторів у системі визначається кількістю підрозділів, які розміщують інформацію. До таких можна віднести деканати факультетів, лігу студентів, профспілку. Ці користувачі мають рівні права, можуть змінювати інформацію тільки у відповідних папках сховища і не мають доступу до папок інших підрозділів.

Для адміністрування цієї системи достатньо одного користувача з відповідними правами.

Одночасно на моніторі відображаються кілька документів у форматі jpg або pdf. Таким чином, обсяг інформації, що відображається на терміналі, не перевищує 100 Мбайт.

Як ресурси для зберігання інформації в даній інформаційній системі використовуються сервери, які розподілені по корпусах. Оскільки обсяг інформації, яку необхідно зберігати на сервері, не перевищує 10 Гбайт (з розрахунку 40 терміналів*100 Мбайт), то достатньо було б використання тільки одного сервера. Однак це повністю суперечить пропонованому підходу зберігання інформації з можливістю постійного доступу до неї. Не можна зберігати інформацію тільки в одному екземплярі, особливо, якщо до неї потрібен

постійний доступ. Тому бажано використовувати 2-3 сервери для зберігання інформації, що виводиться через термінали.

Інформація, що розміщується на інформаційних панелях, може бути досить специфічною і затребуваною тільки студентами одного факультету. Тому зміст інформаційних панелей залежить від місця розташування самого терміналу. Модератори можуть обрати термінали для розміщення інформації з тих, що їм надані в рамках їхніх прав. Наприклад, для деканату права обмежуються його навчальним корпусом. Якщо модератору потрібно розмістити інформацію на термінали, до яких він не має доступу, він зв'язується з головним модератором, який має доступ до всіх терміналів. Така функція може знадобитися на час заселення студентів у гуртожитки, оскільки оголошення про заселення бажано розмістити на кожному терміналі незалежно від його розташування.

Крім місця розміщення інформації модератор має вказати період її актуальності. Після застарівання інформацію видаляють із серверів.

На підприємстві «Комтел» є три дата-центри різної потужності, що розташовані в навчальних корпусах "Л", "М" і "С". Відповідно, сервери для зберігання інформації розташовуватимуться там.

Для кожного терміналу визначено основний сервер, до якого він звертається за інформацією. При цьому термінал зберігає інформацію на своїй карті пам'яті, що дає змогу не передавати інформацію мережею постійно. Термінал повністю оновлює інформацію щодня. Таким чином, застаріла інформація не відображається на терміналі.

Коли модератор визначає термінал для відображення інформації, визначається сервер для запису цієї інформації, а вже після цього інформація копіюється на інші сервери. Тому кожен із серверів зберігає всю інформацію, що виводиться через термінали системи. Це резервне копіювання дає змогу працювати системі інформування в разі виходу з ладу одного або навіть двох серверів. Якщо основний сервер терміналу не працює, то він просто звертається до іншого робочого сервера.

2.5 Забезпечення захисту сховища системі інформування

У сучасному світі існує велика кількість різних алгоритмів шифрування. Деякі з них з'явилися ще до винаходу обчислювальної техніки і є досить простими. Поява ЕОМ і комп'ютерних систем підштовхнула розвиток криптографічних методів. Обчислювальна техніка може обробляти великі обсяги інформації. Тому ускладнилися алгоритми для шифрування та дешифрування інформації, а також з'явилися нові види атак на шифр. Атака на шифр (криптографічний аналіз) - це процес дешифрування закритої інформації без знання секретного ключа, а іноді й алгоритму шифрування [25].

Складність злому алгоритму шифрування визначається його криптографічною стійкістю.

Криптографічна стійкість - характеристика ефективності шифру, що визначає його стійкість для дешифрування шляхом атаки. Криптографічна стійкість вимірюється часом або вартістю засобів, необхідних для отримання секретної інформації за умови, що не відомий ключ шифрування [26].

Зберегти в секреті широко використовуваний алгоритм практично неможливо, виходячи з цього, сформульовано правило Керкгоффа:

"Під час оцінки надійності шифрування необхідно припускати, що зловмисник знає про використовувану систему шифрування все, окрім застосовуваного ключа. " Таким чином, стійкість шифрування обумовлюється надійністю його ключа [27].

Підвищення криптографічної стійкості системи можливе двома способами:

- ускладнення алгоритму шифрування;
- збільшення секретного ключа.

За відсутності інформації, крім зашифрованого тексту, криптографічний аналіз вимагає багато ресурсів, оскільки в цьому випадку необхідний повний перебір усіх можливих ключів. Однак, такий варіант захисту рідко можливий. Зазвичай, у криптографічного аналітика є додаткова інформація, яка спрощує

процес криптографічного аналізу. Знання ключа й алгоритму шифрування дає змогу перетворювати будь-який зашифрований текст на відкритий.

Разом із цим, наявність відкритого тексту суттєво спрощує завдання отримання ключа шифрування. Зокрема, у тексті великого обсягу досить часто зустрічаються повторення символів. Якщо зашифрований текст утворено за алгоритмом простої заміни з використанням короткого ключа, то схожі повторення символів спостерігатимуться в зашифрованому тексті. Виявлення цих закономірностей сприяє отриманню секретного ключа і значно скорочує використання ресурсів для криптографічного аналізу.

Можна виділити два види алгоритмів шифрування інформації:

- симетричні;
- асиметричні.

Оскільки дана система є системою інформування співробітників, мета її функціонування - поширення інформації по корпусах підприємства «Комтел». Тому гостро стоїть проблема можливості поширення некоректної або неприпустимої інформації. Таким чином, захист інформаційної системи будується на захисті інформації від підміни.

Для цього в системі застосовується ідентифікація та автентифікація, розмежування доступу та шифрування інформації.

Процес ідентифікації користувача визначає, до якої групи він належить, а разом із цим і його права на доступ до сховища інформації в системі.

Під час додавання файлів модератором вони зашифровуються і надсилаються в сховище. При підключенні безпосередньо до сховища, інформація не піддається використуваному криптографічному перетворенню. Тому під час передачі цієї інформації на термінал зворотне криптографічне перетворення зробить її недоступною для прочитання. У підсумку, для забезпечення інформаційної безпеки потрібно забезпечити збереження секретного ключа.

Поділ потоків інформації дає змогу використовувати асиметричні схеми шифрування інформації.

Асиметричні алгоритми шифрування у своїй роботі використовують відкриті ключі для шифрування і закритий ключ для дешифрування. Асиметричні алгоритми побудовані на ідеї використання односторонніх функцій "із секретом". Такою функцією є $y=f(x)$, значення якої легко обчислити. Зворотне перетворення виконати складно, якщо не знати "секрету", тобто закритого ключа.

Стійкість більшості сучасних асиметричних алгоритмів ґрунтується на двох математичних проблемах:

- дискретне логарифмування в скінченних полях;
- факторизація великих чисел.

Розв'язання цих проблем потребує великих витрат обчислювальних ресурсів, оскільки не мають ефективних алгоритмів розв'язання і важко обчислюються навіть методом "грубої сили".

Найпоширенішим алгоритмом шифрування з відкритим ключем є RSA. Стійкість RSA ґрунтується на складності факторизації великих чисел. Крім шифрування/дешифрування, цей алгоритм може бути використаний для генерації та перевірки електронного цифрового підпису [28].

Генерація ключів проводиться учасниками обміну за такими правилами:

- вибирають два великих цілих числа p і q ;
- обчислюється модуль системи $n = p * q$ і $\varphi = (p - 1)(q - 1)$ - функція Ейлера;
- вибирається досить велике число e таке, що:
- $1 < e < \varphi(n)$ і взаємно просте з $\varphi(n)$;
- використовуючи розширений механізм Евкліда, обчислюється велике число d , що відповідає умові: $ed = 1 \pmod{\varphi(n)}$, де $1 < d < \varphi(n)$.

Збільшення розряду чисел p і q веде до підвищення стійкості алгоритму, а також веде до уповільнення операції перетворення. З огляду на це обирається певний компроміс залежно від можливостей і вимог системи.

Секретним ключем є пара чисел (n, d) , а відкритим ключем - пара чисел (n, e) . Після вибору параметрів (n, e, d) , можна передавати зашифровані повідомлення.

Їх передача складається з декількох кроків:

- вхідне повідомлення розбивається на блоки m_i , їхній розмір визначається за допомогою цілого k , що задовольняє нерівність $10k-1 < n < 10k$;
- обчислюється $ci = te \bmod n$;
- значення ci , що є зашифрованим повідомленням, відсилається відкритими каналами зв'язку.

Дешифрування полягає в обчисленні $mi = cd \bmod n$.

Цей алгоритм застосовується для шифрування в системі інформування. Модератори використовують ключі для шифрування, а клієнтські термінали використовують ключі для дешифрування. Причому ключі модераторів не можна повною мірою назвати "відкритими", оскільки їх формування відбувається з файлу, який не перебуває у відкритому доступі.

Модератор під час додавання інформації використовує ключ, що формується за певним алгоритмом перед шифруванням. (Рис. 2.7) Для цього в кожного модератора є набір файлів для формування ключів, що зберігається на USB-пристрої в модератора і використовується під час додавання інформації до сховища. Відповідно відсутність USB-пристрою з файлами не дасть змоги користувачеві додати інформацію до сховища.



Рисунок 2.7 - Завантаження файлу в сховище

Оскільки ця система є системою інформування, а не системою екстреного оповіщення, то не потрібне миттєве оновлення виведеної інформації. Інформація,

по суті, є досить статичною і оновлюється кілька разів на день. Тому, незважаючи на обмежені обчислювальні потужності термінального пристрою, дешифрування відбувається безпосередньо на терміналі перед виведенням інформації.

Кожен термінал, що відображає інформацію, має власний ключ шифрування, відмінний від інших терміналів. У програмі дешифрування зберігається алгоритм для формування ключа із зовнішнього файлу, який знаходиться тільки на одному терміналі. Тому за загальним алгоритмом формуються різні ключі залежно від терміналів.

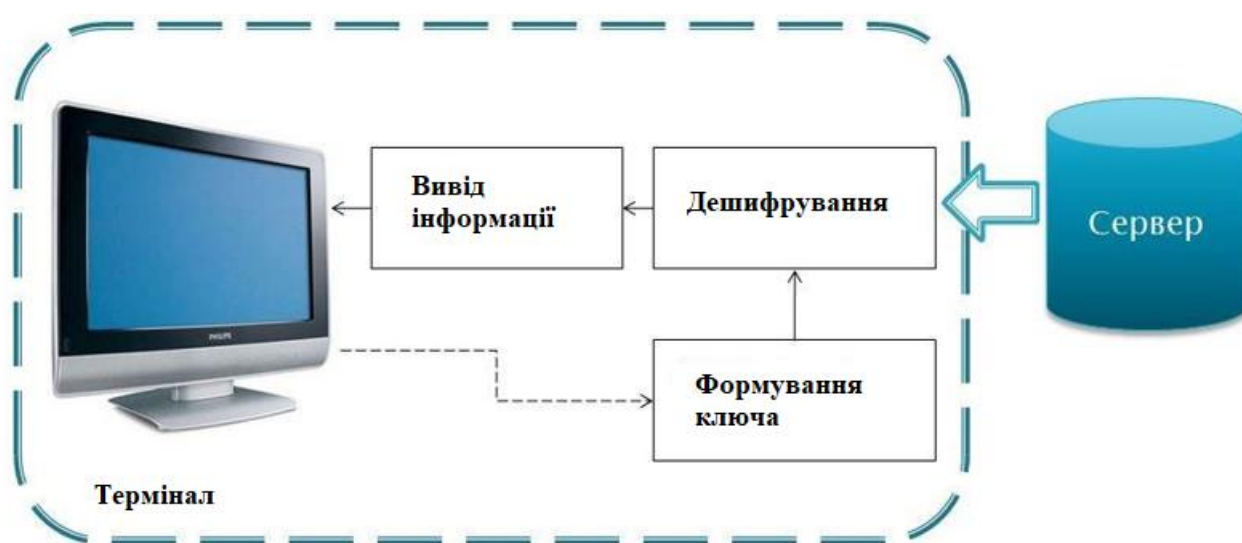


Рисунок 2.8 - Виведення інформації на термінал.

Таке формування ключів пов'язане з тим, що використання спільного ключа збільшує ймовірність повного виведення з режиму нормальної роботи всієї системи в разі отримання цього ключа зломщиком. Якщо ж використовуються різні ключі, отримання одного ключа і злом терміналу виводить з ладу тільки конкретний термінал, а інша система продовжує працювати.

Якщо ключ було підібрано, адміністратор формує нові ключі для терміналу і модератора, шляхом створення нового вихідного файлу ключа і замінює його на терміналі, а також передає новий ключ модератору.

Специфіка цієї системи в тому, що втрата інформації або її поширення не є критичним. Тому мета шифрування інформації - захист від підміни та виведення

недостовірної і некоректної інформації. Навіть виведення з ладу одного або декількох термінальних пристроїв, шляхом видалення на них файлів для формування ключів, призведе тільки до того, що на термінал не виводитиметься нова інформація зі сховища. На відновлення роботи терміналу адміністратору знадобиться не більше однієї години.

Система інформування є розподіленою інформаційною системою, причому розподілені не тільки "клієнти", а й сервери для зберігання інформації.

Таким чином, для сховища представленої інформаційної системи потрібна наявність таких особливостей:

- організація сховища як хмарного сервісу;
- користувацький інтерфейс;
- система ідентифікації користувачів;
- розмежування доступу до сховища інформації;
- шифрування збереженої інформації.

3 ОРГАНІЗАЦІЯ ЗАХИЩЕНОГО ХМАРНОГО СХОВИЩА

3.1 Структура та організація сховища

Сховище організовано у вигляді ієрархічної структури каталогів, що містять файли. Поділ директорій у сховищі ґрунтується на фізичному розташуванні терміналів, що виводять інформацію. На першому рівні йдуть директорії, що відповідають корпусам підприємства «Комтел». (Рис. 3.1) У кожній із цих директорій містяться піддиректорії конкретних терміналів, розташованих у цьому корпусі. Відповідно до прав кожного модератора, сховище відкривається для нього як та директорія, до якої він має доступ. Повністю все сховище може бачити тільки головний модератор системи інформування, оскільки тільки він має права на додавання інформації на будь-який термінал і, отже, у будь-який каталог сховища

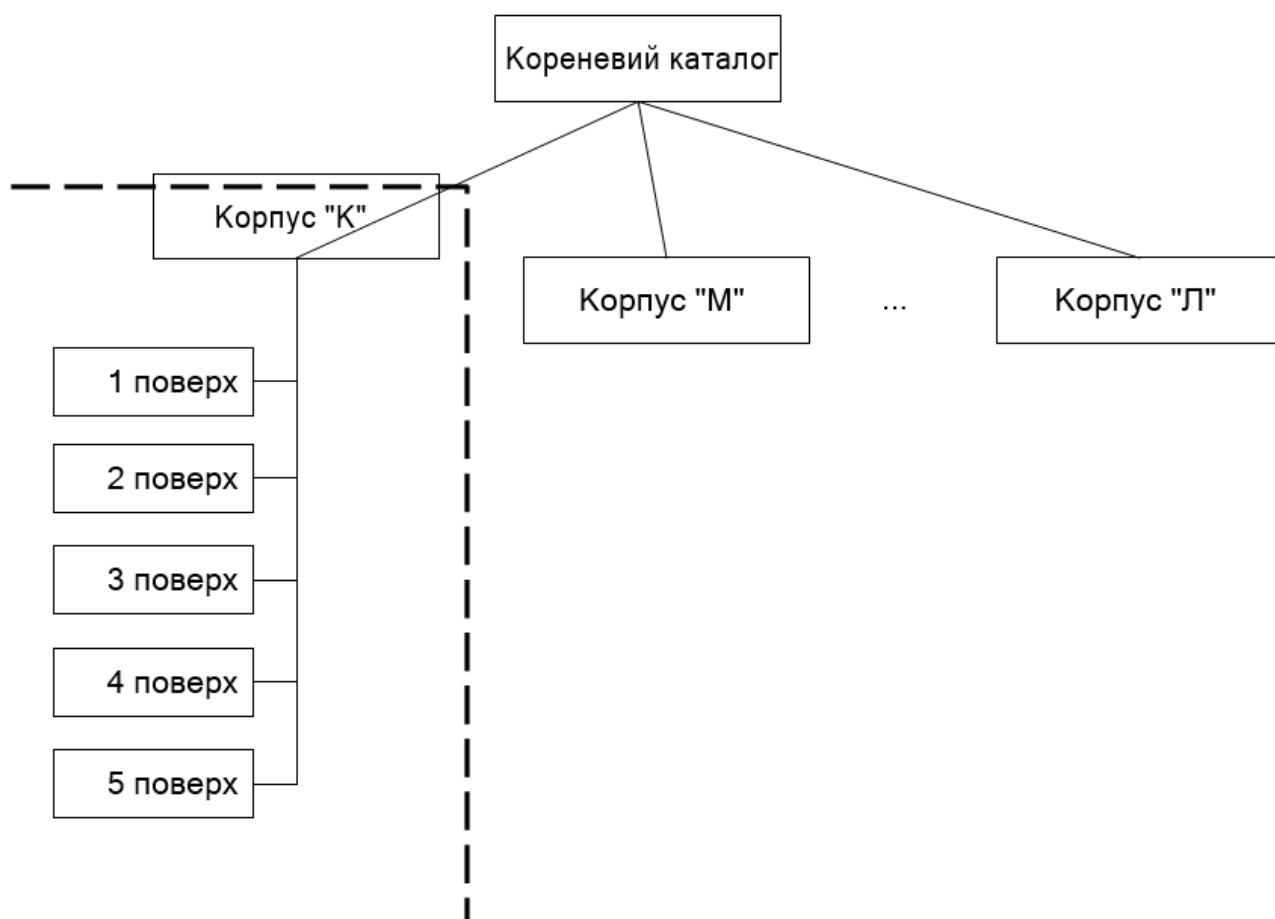


Рисунок 3.1 - Ієрархічна структура сховища

Розглянемо розмежування доступу на прикладі директорії, в якій міститься інформація для терміналів корпусу "К". У цьому корпусі розташовуються фізико-технічний і хімічний факультети. Тому модератори системи інформування від деканатів цих факультетів працюють із директорією терміналів корпусу "К". Таким чином, вони бачать у "своєму" сховищі директорії, в яких міститься інформація для терміналів, розташованих з першого по п'ятий поверх корпусу, і можуть завантажувати інформацію на ці термінали. У корпусі "К" є кілька приміщень, які теж можуть виводити інформацію через систему інформування. Однак для модератора доступний тільки один термінал. У цьому разі термінал для виведення інформації вибирається відповідно до розташування.

Такий поділ рівня доступу користувачів насамперед пов'язаний зі специфічністю розміщуваної ними інформації. Крім того, це дає змогу зменшити кількість користувачів, які зберігають файли для створення ключів шифрування, до кількох терміналів. У підсумку, в системі є один користувач, який має доступ до всіх директорій у сховищі, дванадцять модераторів, які мають доступ до директорії, відповідної корпусу його розташування, а також багато користувачів, які мають доступ тільки до однієї директорії.

3.2 Програма для модератора системи інформування

Як частину системи сховища реалізовано програму клієнтського пристрою модератора. Програма має запускатися на комп'ютері під управлінням операційної системи Windows. Ця умова ґрунтується на тому, що для додавання файлів у сховище користувачі-модератори використовуватимуть стаціонарний персональний комп'ютер, що знаходиться на робочому місці.

Програмне забезпечення, необхідне для функціонування програми, має містити в собі:

- ОС Windows 2000 і вище;
- Windows Installer 3.0;
- Microsoft .NET Framework 2.0 (DotNetFix).

"Програма для модератора інформаційної системи" написана на мові C#, що входить до складу продукту Microsoft Visual Studio та має платформу .NET.

Програма призначена для завантаження файлів у сховище інформаційної системи з метою подальшого виведення інформації на термінали, розміщені в корпусах підприємства «Комтел».

Опис логічної структури

"Програма для модератора інформаційної системи" ґрунтується на структурі сховища інформації, представленій в розділі 2.

Виклик і завантаження

"Програма для модератора інформаційної системи запускається шляхом відкриття файлу Filestorage.exe.

Вхідні дані

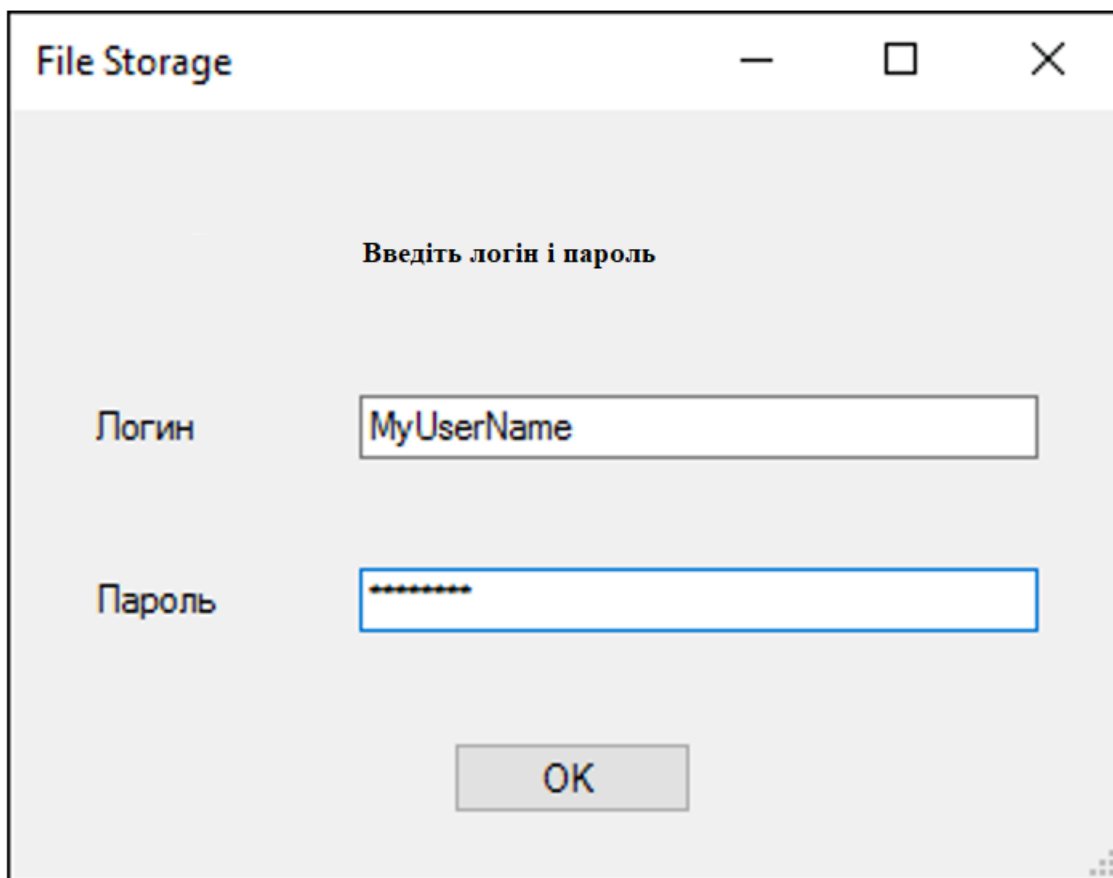
Файли, обрані модератором для завантаження на сервер; файли для формування ключа шифрування.

Вихідні дані

Файл, що пройшов алгоритм шифрування.

Стартове вікно "програми для модератора інформаційної системи" представлено вікном ідентифікації користувача.

Перед відкриттям сховища користувач має пройти процедури ідентифікації та аутентифікації. У стартовому вікні користувач бачить напис із проханням ввести логін і пароль для отримання доступу. (Рис. 3.2) Для отримання значень використовуються текстові поля введення. Значення, введене в поле "Логін", доступне для читання користувачем. Значення, що вводиться в поле "Пароль" не відображається в явному вигляді і кожен символ представлений символом "*". Таке відображення є одним із способів захисту пароля модератора від розповсюдження і подальшого злому облікового запису модератора.



File Storage

Введіть логін і пароль

Логин MyUserName

Пароль *****

OK

Рисунок 3.2 - Стартове вікно для ідентифікації користувача.

Алгоритм роботи модуля представлено на рис. 3.3. Користувач вводить свій логін і пароль. Для значення, введеного в поле "пароль", обчислюється хеш-функція за алгоритмом MD5. Це обумовлено тим, що пароль не зберігається у відкритому вигляді, а обчислюється за таким самим алгоритмом. Значення "логін" і обчислений хеш значення "пароль" звіряються з базою даних користувачів. У разі підтвердження логіна і пароля користувач потрапляє в основний модуль програми і може працювати зі сховищем інформації. Якщо ж "логін" і "пароль" введено неправильно, користувачеві виводиться повідомлення про помилку, і він заново може пройти процедуру ідентифікації.



Рисунок 3.3 – Блок-схема модуля ідентифікації

Модератори мають право тільки додавати файли до сховища, а також видаляти свої раніше завантажені файли. Тому запис файлів у сховище здійснюватиметься за таким алгоритмом. (Рис. 3.4) Неформальний опис алгоритму для запису файлів у сховище:

- формування модератором інформації для додавання в сховище;
- перевірка наявності у модератора ключа для шифрування; якщо ключ доступний, то п.3, інакше п.5;
- виконання модуля шифрування;

- надсилання зашифрованих файлів у сховище;
- вихід.

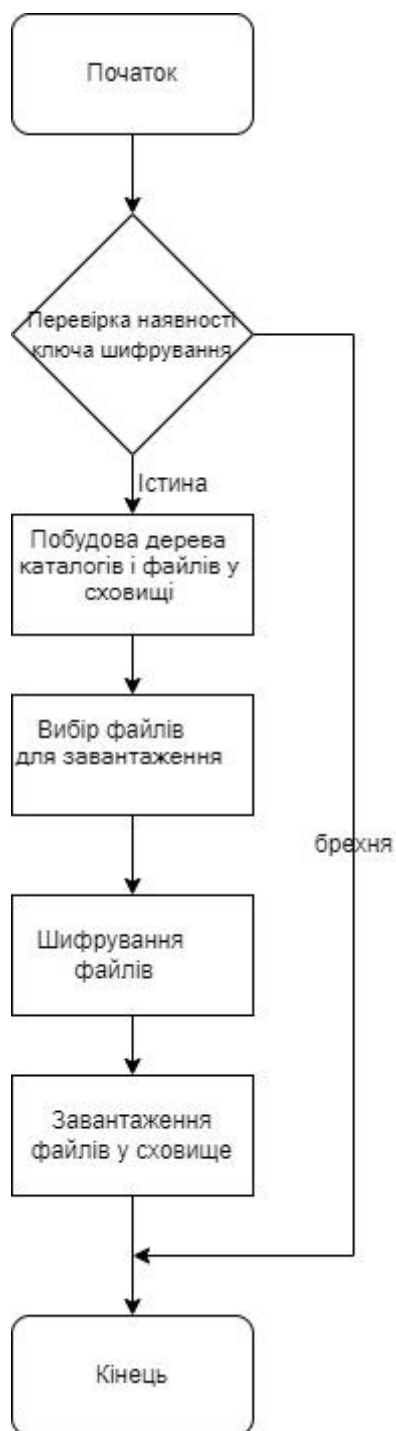


Рисунок 3.4 - Блок-схема основної програми для модератора

Після проходження процедури ідентифікації у користувача перевіряється наявність файлу для створення секретного ключа. Для відкриття основного вікна програми відбувається побудова дерева каталогів сховища. Модератор вибирає

шлях до сховища, а потім вказує файли для завантаження зі свого комп'ютера. Перед завантаженням файлів викликається модуль для їх шифрування. Зашифровані файли завантажуються на сервер.

У вікні основної програми користувач бачить дерево файлів і каталогів, розташованих на сервері. (Рис. 3.5) Під час натискання кнопки "Загрузить" відкривається допоміжне вікно, що дає змогу користувачеві вибрати файли зі свого комп'ютера для завантаження на сервер. Кнопка "Видалити" служить для видалення файлів користувача, які більше не потрібно виводити на термінали з якихось причин, наприклад через зміну цієї інформації або втрату її актуальності.

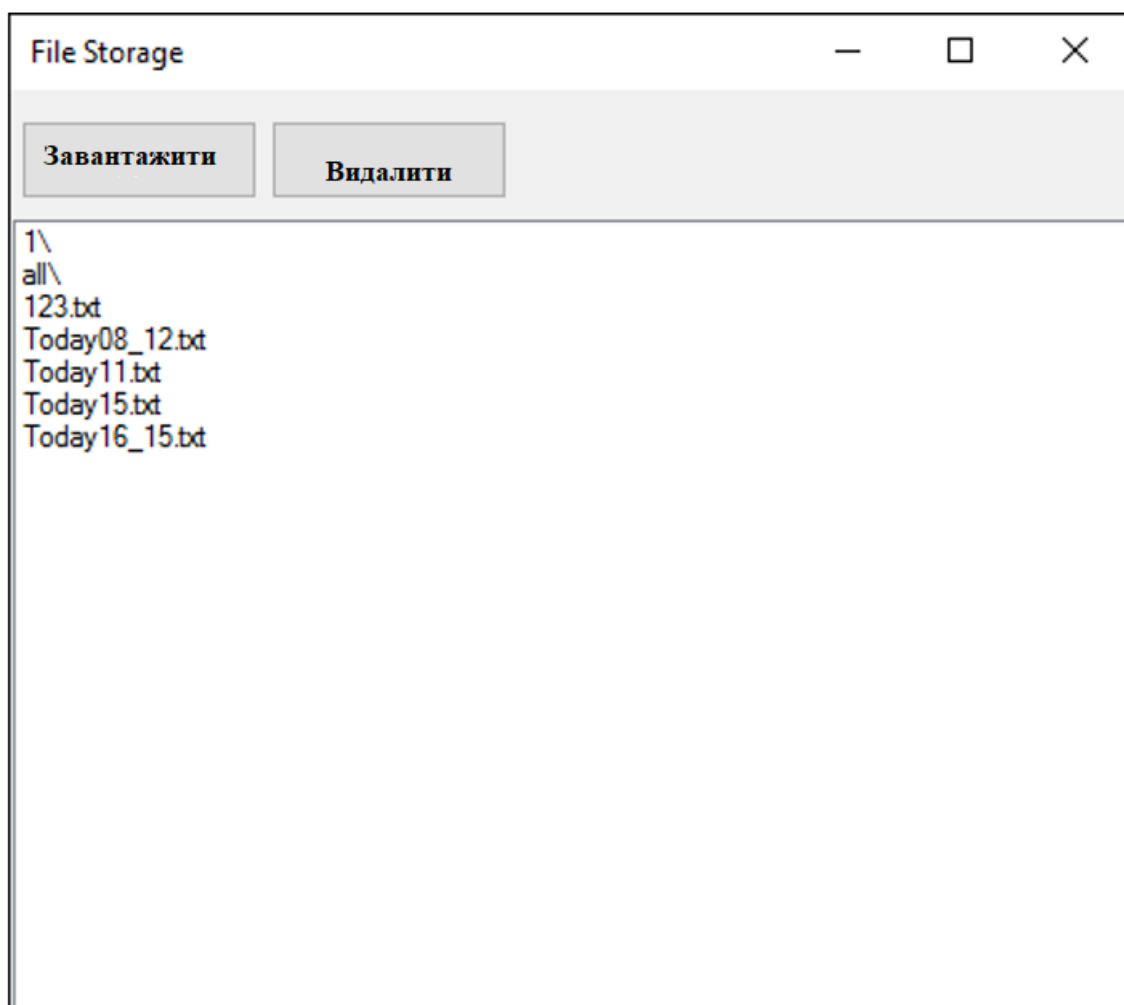


Рисунок 3.5 - Основне вікно програми

Алгоритм шифрування має забезпечувати таку криптографічну стійкість, щоб час злому, необхідний для отримання конкретної інформації, був більшим за час актуальності цієї інформації.

Шифрування в системі RSA здійснюється шляхом зведення вихідного тексту X

у ступінь відкритого ключа КО за модулем великого числа r : $Y = E_{КО}(X) = X^{КО} \bmod r$.

Дешифрування здійснюється шляхом зведення шифротексту Y у ступінь закритого ключа КС за модулем r :

$$Y^{КС} \bmod r = (X^{КО} \bmod r)^{КС} \bmod r = X^{КО \cdot КС} \bmod r = X \bmod r.$$

Можливість отримання вихідного тексту із зашифрованого забезпечується властивістю відкритого і закритого ключів, для яких має виконуватися рівність

$$(КО \cdot КС) \bmod \varphi(r) = 1, \text{ де } \varphi(r) - \text{функція Ейлера від } r.$$

Закритий ключ є мультиплікативним інверсним за модулем $\varphi(r)$ для відкритого. За теоремою Ейлера

$$a^{\varphi(r)} \equiv 1 \bmod r, \text{ де } (a, r) = 1.$$

Відомо, що якщо $a \equiv b \bmod r$, то $a^n \equiv b^n \bmod r$. Із цього випливає, що $X^{КО \cdot КС} \equiv X \bmod r$.

Оскільки для модулярної арифметики справедливо $m - a \equiv m - b \bmod r$, то ми можемо

$$\text{помножити ліву і праву частини рівності на } X: X^{КО \cdot КС} \equiv X \bmod r.$$

Якщо уявити показник ступеня $КО \cdot КС$ як добуток значень КО і КС, то отримаємо $КО \cdot КС \equiv 1 \bmod \varphi(r)$.

Блок-схему програми шифрування за алгоритмом RSA наведено на рис. 3.6.

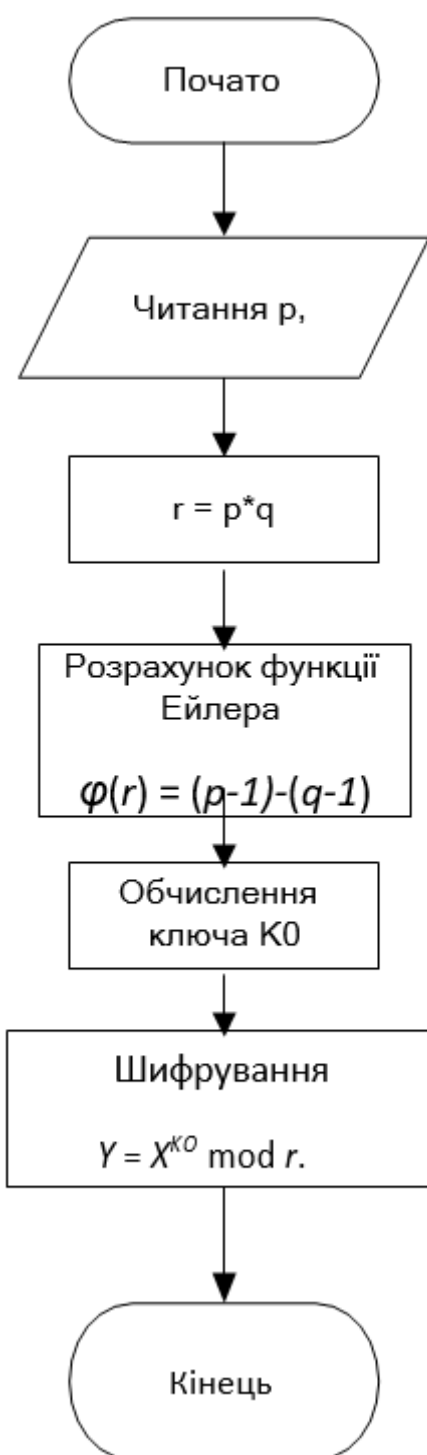


Рисунок 3.6 - Блок-схема модуля шифрування

Процедура формування параметрів алгоритму RSA має такий вигляд:

- з файлу для формування ключів зчитуються два великих простих числа p і q ;
- обчислюється добуток цих чисел: $r = p \cdot q$;
- знаходиться функція Ейлера від r : $\varphi(r) = (p-1) \cdot (q-1)$;

– генерується значення відкритого ключа КО, виходячи з таких умов:
 $KO < \varphi(r)$, $(КО, \varphi(r)) = 1$;

Далі файл шифрується з використанням отриманих параметрів. Для виконання процедур шифрування і дешифрування використовується алгоритм швидкого піднесення до степеня за модулем, що дає змогу для додатних цілих a , z і n обчислити $x = az \bmod n$. Лістинг коду функції для обчислення за алгоритмом швидкого піднесення до степеня наведено на рис. 3.7.

```
unsigned long long degree(unsigned long long a, unsigned long long z, unsigned long long n) {
    unsigned long long count = 1;
    if (!z) return 1;
    while (z) {
        if (z%2==0)
        {
            z/=2;
            a*=a;
            a%=n;
        }
        else
        {
            z--;
            count *= a;
            count %= n;
        }
    }
    count %= n;
    return count;
}
```

Рисунок 3.7. Лістинг коду функції швидкого піднесення до степеня

Початкове значення змінної z перевіряється на парність. Якщо z парне значення, то z ділиться на 2, значення змінної a підводиться до квадрата і знаходиться його залишок від ділення на n . Інакше, якщо z непарне, то значення z зменшується на 1, змінна $count$, що є значенням функції, що повертається, множиться на a , і від отриманого значення береться залишок від ділення на n . Цей цикл повторюється доти, доки значення z не стане рівним нулю.

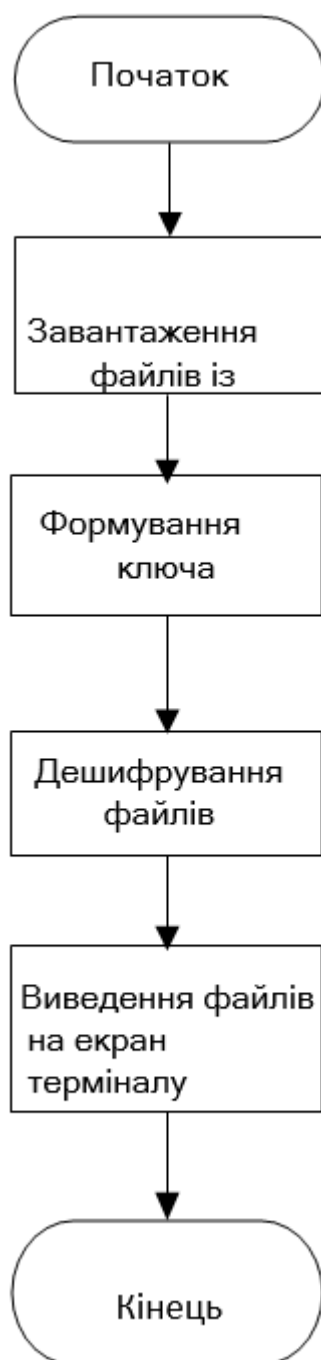
Отриманий файл записується на сервер сховища системи інформування.

3.3 Модуль програми для клієнта-терміналу

Клієнтський термінал має можливість тільки зчитувати файли зі сховища. Тому доступ до сховища завжди здійснюється одноманітно. Неформальний опис алгоритму доступу до інформації через клієнтський термінал:

- запит терміналу на зчитування файлів;
- перевірка наявності доступу; якщо доступ підтверджено, то п.3, інакше п.6;
- читання файлів зі сховища;
- почергове дешифрування файлів за допомогою програми;
- виведення інформації на екрані терміналу;
- вихід.

Для зворотного перетворення файлу, з метою отримання інформації, використовується програма дешифрування. На рис. 3.8 представлено схему роботи програми дешифрування файлів для відображення на терміналі.



Риунок 3.8 Блок-схема програми терміналу

Процедура формування параметрів алгоритму RSA для терміналу має такий вигляд:

- З файлу для формування ключів зчитуються два великих простих числа: p і q , $p \neq q$;
- обчислюється добуток цих чисел: $r = p \cdot q$;
- знаходиться функція Ейлера від r : $\phi(r) = (p-1) \cdot (q-1)$;

– генерується значення відкритого ключа КО, виходячи з таких умов:
 $КО < \varphi(r)$, $(КО, \varphi(r)) = 1$;

– обчислюється мультиплікативне інверсне за модулем $\varphi(r)$ для КО.
 Отримане значення буде закритим ключем КС.

Програма зчитує інформацію із зашифрованого файлу. Функція дешифрування виконує зворотнє перетворення інформації. Наприкінці відкрита інформація записується у вихідний файл, який у такому вигляді може бути виведений на термінал.

Для обчислення мультиплікативного інверсного використовується розширення алгоритму Евкліда (рис. 3.9).

```
unsigned long long mul_mod_rev(unsigned long long x, unsigned long long mod) {
    long long M_MUL = mod;

    long long g0, g1, g2, v0, v1, v2, y;

    if(x <= 1) return x;

    g0 = mod; g1 = x; g2 = 0;
    v0 = 0; v1 = 1; v2 = 0;

    while(g1 != 0) {
        y = g0 / g1;
        g2 = g0 - y * g1;
        v2 = v0 - y * v1;
        g0 = g1; g1 = g2;
        v0 = v1; v1 = v2;
    }
    if(v0 < 0) v0 = v0 + mod;
    return ((unsigned long long)v0);
}
```

Рисунок 3.9 - Лістинг коду для розрахунку ключа

Розширений алгоритм Евкліда дає змогу знайти значення x_1 і y_1 , за яких виконується рівність $x_1 \cdot a + y_1 \cdot b = d_1$, де $d_1 = \text{НОД}(a, b)$. Якщо $a > b$ і числа a, b - взаємно прості, тобто $d_1 = 1$, то y_1 є мультиплікативним інверсним для b за модулем a . [29]

Якщо розширений алгоритм Евкліда використовується для обчислення мультиплікативного інверсного i в результаті виконання алгоритму отримано від'ємне значення, то до нього необхідно додати величину a .

Таким чином, було розроблено структуру організації сховища системи інформування. У середовищі розробки Microsoft Visual Studio 2013 розроблено програму для модераторів системи інформування та запропоновано її реалізацію мовою C#. Реалізація модуля шифрування та дешифрування мовою C дає змогу використовувати його для будь-яких операційних систем, зокрема Linux і подібних.

ВИСНОВКИ

Аналіз актуальних на сьогодні рішень у сфері побудови сховищ для інформаційних систем, як з апаратного, так і з програмного погляду, дав змогу виявити переваги та недоліки цих рішень і технологій. На підставі пред'явлених вимог було спроектовано систему зберігання інформації на підприємстві. Спроектоване сховище задовольняє таким вимогам: зберігання різномірної інформації на власних ресурсах, з можливістю поділу користувачів за рівнем доступу до даних, а також можливість роботи користувача зі своїми даними з будь-якого комп'ютера всередині мережі підприємства.

Реалізацію хмарного сховища інформації підприємства виконано для конкретної інформаційної системи, що є системою інформування підприємства «Комтел». Доступ до сховища для користувачів розмежований, залежно від їхніх прав і ролі в системі. Програмні додатки розроблено в середовищі Microsoft Visual Studio 2013 з використанням мови програмування високого рівня C#. Підпрограма для шифрування та дешифрування використовує алгоритм RSA і розроблена мовою C. Лістинг програми наведено в ДОДАТКУ А.

Під час цієї роботи зроблено таке:

- спроектовано хмарне сховище підприємства;
- розподілені ролі користувачів;
- визначено методи захисту сховища підприємства;
- обрано алгоритм шифрування інформації;
- розроблено програми для доступу клієнтських пристроїв у сховище.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Storage Technologies Overview. [https://technet.microsoft.com/en-us/library/dn610883\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/dn610883(v=ws.11).aspx) (Дата звернення: 01.05.2023)
2. Проектування та розробка корпоративних інформаційних систем - Режим доступу: <http://citforum.com/cfin/prcorpsys/index.html> (Дата звернення: 01.05.2023)
3. Методи побудови систем зберігання даних. <http://citforum.com/hardware/data/db/> (Дата звернення: 01.05.2023)
4. "Хмарні сервіси" - що це таке. <http://sd-company.su/article/cloud/service> (Дата звернення: 01.05.2023)
5. Хмарні обчислення (світовий ринок). <http://www.tadviser.com/index.php/> (Дата звернення: 01.05.2023)
6. Що таке PaaS? <https://azure.microsoft.com/overview/what-is-paas/> (Дата звернення: 01.05.2023)
7. SaaS - що це таке? Software as a Service - програмне забезпечення як послуга. <https://azure.microsoft.com/overview/what-is-saas/> (Дата звернення: 01.05.2023)
8. Cloud Computing: що таке інфраструктура як послуга. <https://technet.microsoft.com/en-us/library/hh509051.aspx> (Дата звернення: 01.05.2023)
9. Знайомимося з SecaaS - переваги безпеки з хмари. [https://www.anti-malware.ru/analytics/Technology_Analysis/ SecaaS_security_cloud_computing](https://www.anti-malware.ru/analytics/Technology_Analysis/SecaaS_security_cloud_computing)
10. Девід Чеппелл. Знайомство з Microsoft Azure. <https://msdn.microsoft.com/ru-ru/jj677178.aspx> (Дата звернення: 01.05.2023)
11. Bluemix відкриває шляхи до PaaS/ <https://www.ibm.com/developerworks/ru/library/cl-bluemix-summary/> (Дата звернення: 01.05.2023)
12. Акрітас А. Основи комп'ютерної алгебри з додатками: Пер. з англ. - К.,

Мир, 1994. - 544 с.

13. Клас MD5

[https://msdn.microsoft.com/ruru/library/system.security.cryptography.md5\(v=vs.110\).aspx](https://msdn.microsoft.com/ruru/library/system.security.cryptography.md5(v=vs.110).aspx) (Дата звернення: 01.05.2023)

14. Visual C#. [https://msdn.microsoft.com/ruru/library/kx37x362\(v=vs.120\).aspx](https://msdn.microsoft.com/ruru/library/kx37x362(v=vs.120).aspx) (Дата звернення: 01.05.2023)

15. Хмарна синхронізація/ <https://www.stekspb.ru/autsorsing-it-infrastruktury/articles/it-glossary/cloud-sync/> (Дата звернення: 01.05.2023)

Додаток А

Лістинг А.1 - Код програми

```

namespace MyStore
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        Форма renForm =
        new Form2();
        Форма stForm =
        new
        StartForm();
        private string[] getContent(string path)
        {
            int i,
            dirsCount,
            filesCount;
            string[]
            files, dirs,
            totalContent;
            path =
            String.Concat(
            path, "\\");
            dirs =
            Directory.GetDirec
            tories(path);
            dirsCount =
            dirs.Count();
            for (i = 0; i < dirsCount; i++)
            {
                dirs[i] = dirs[i].Replace(path, "");
                dirs[i] = string.Concat(dirs[i], "\\");
            }
            files =
            Directory.GetF
            iles(path);
            filesCount =
            files.Count();
            for (i = 0; i < filesCount; i++)

```

```

        files[i] = files[i].Replace(path, "");
    }
    totalContent = new
    string[filesCount +
    dirsCount]; for (i = 0; i
    < dirsCount; i++)
        totalContent
        [i] =
        dirs[i]; for
        (i = 0; i <
        filesCount;
        i++)
            totalContent[i +
            dirsCount] =
            files[i]; return
            totalContent;
    }

private void fillContent(string path)
{
    string
    RootDirectory =
    "D:\\\\MyStore\\";
    string[] content;
    int i, contentCount;
    //ContentBox.Directo
    ry = RootDirectory;
    content =
    getContent(path);
    ContentBox.Items.Cle
    ar();
    if (path !=
        RootDirectory
        )
        ContentBox.It
        ems.Add("..."
        );
    currentDir.Text
    = path;
    contentCount =
    content.Count()
    ; for (i = 0; i
    < contentCount;
    i++)
        ContentBox.Items.Add(content[i].ToString());
    }

private void copy(string src, string dst)
{
    DialogRe
    sult
    result;

```

```

if (src
!= dst)
{
    if (Directory.Exists(src))
    {
        if (Directory.Exists(dst))
        {
            result = MessageBox.Show("Така директорія вже є.
Замінити файли при збігу?", "Підтвердження заміни",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (result == DialogResult.Yes)
            {
                copyDir(src, dst, true);
            }
        }
        іще
        {
            copyDir(src, dst, false);
        }
    }
    іще
    {
        if (File.Exists(src))
        {
            if (File.Exists(dst))
            {
                result = MessageBox.Show("Такий файл уже є.
Замінити?", "Підтвердження заміни", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);
                if (result == DialogResult.Yes)
                { //якщо вирішили замінювати
                    File.Delete(dst);
                        //видаляємо
                    існуючийFile.Copy(src,);
                        //ікопіюємо
                }
            }
            іще
            {
                //якщоні, то
                просто копіюємо
                File.Copy(src, dst); //то
                просто копіюємо
            }
        }
    }
}

```

```

    }
}
іще
{//якщо ні чим не є
    MessageBox.Show(string.Format("{0}      не
                                знайдено",      src),
                                "Помилка",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
іще
{
    MessageBox.Show(string.Format("Початковий і
кінцевий шляхи не повинні збігатися"), "Помилка",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void copyDir(string src, string dst, bool replace)
{
    int
    dirsCount,
    filesCount,
    i; string
    tempDst;
    string[] dirs, files;

    dirs =
    Directory.GetDire
    ctories(src);
    files =
    Directory.GetFile
    s(src); dirsCount
    = dirs.Count();
    for (i = 0; i < dirsCount; i++)
    {
        dirs[i] = dirs[i].Replace(src, "");
        dirs[i] = string.Concat(dirs[i], "\\");
    }

    filesCount =
    files.Count()
    ; for (i = 0;
    i <
    filesCount;
    i++)
        files[i] = files[i].Replace(src, "");
}

```

```

if (!Directory.Exists(dst))
    Directory.CreateDirectory(dst);
foreach (string file in files)
{
    tempDst = string.Concat(dst, file);
    if (File.Exists(tempDst))
    {
        if (replace)
        {
            File.Delete(tempDst);
            File.Copy(string.Concat(src, file),
                tempDst);
        }
    }
    іще
    {
        File.Copy(string.Concat(src, file), string.Concat(dst,
            file));
    }
}
foreach (string dir in dirs)
{
    copyDir(string.Concat(src, dir), string.Concat(dst, dir),
        replace);
}
private void deleteItem(string src)
{
    DialogResult result;
    if (Directory.Exists(src))
    {
        result = MessageBox.Show(string.Format("Ви дійсно хочете
        видалити {0}", src), "Підтвердження видалення",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
        if (result ==
            DialogResult
                .Yes)
            deleteDir(sr
                c);
    }
    іще

```

```

    {
        if (File.Exists(src))
        {
            result = MessageBox.Show(string.Format("Ви дійсно хочете видалити {0}", src), "Підтвердження видалення", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
            if (result == DialogResult.Yes)
            {
                File.Delete(src);
            }
        }
        іще
        {
            MessageBox.Show(string.Format("{0} не знайдено", src), "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

private void deleteDir(string src)
{
    string[] dirs, files;
    dirs =
    Directory.GetDire
    ctories(src);
    files =
    Directory.GetFile
    s(src); foreach
    (string file in
    files)
    {
        File.Delete(file);
    }
    foreach (string dir in dirs)
    {
        deleteDir(dir);
    }
    Directory.Delete(src);
}

private void Form1_Load_1(object sender, EventArgs e)
{

```

```

stForm.ShowDialog();
if (Options.pastrue)
fillContent("D:\\\\MyStor
e\\"); else
{
    MessageBox.Show("Для входу введіть
    вірний логін і пароль");
    this.Close();
}
}

private void ContentBox_MouseDoubleClick(object sender,
MouseEventArgs e)
{
    спробувати
    {
        string
        head,
        tail,
        path; head
        =
        currentDir
        .Text;
        //drive =
        Directory.GetDirector
        yRoot(head); if
        (ContentBox.SelectedI
        tem != null)
        {
            tail = ContentBox.SelectedItem.ToString();
            if ((ContentBox.SelectedIndex == 0) && (tail == "..."))
            {
                path = head.Remove(head.Length - 1);
                path = path.Remove(path.LastIndexOf('\\') + 1);
            }
            іще
            {
                path = String.Concat(head, tail);
            }
            if (Directory.Exists(path))
            {
                fillContent(path);
            }
        }
    }
}

```

```

        іще
        {
            if
                (File.Exists(path
                ))
                Process.Start(pat
                h);
            іще
                MessageBox.Show(path, "Невірний шлях");
        }
    }

    ContentBox.SelectedIndex = -1;
}

catch (System.Exception ex)
{
    MessageBox.Show(ex.Message, "Помилка",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void DownloadButton_Click(object sender, EventArgs
e)
{
    Форма OpenForm
    = new Form4();
    OpenForm.ShowDialog();
    if (Options.diropen)
    {
        string
        dst;
        try
        {
            string fileName =
            Path.GetFileName(Options.Op
            Dir); dst =
            string.Concat(currentDir.Te
            xt, fileName);
            copy(Options.OpDir, dst);
            fillContent(currentDir.Text
            );
        }
        catch (Exception ex)
        {

```

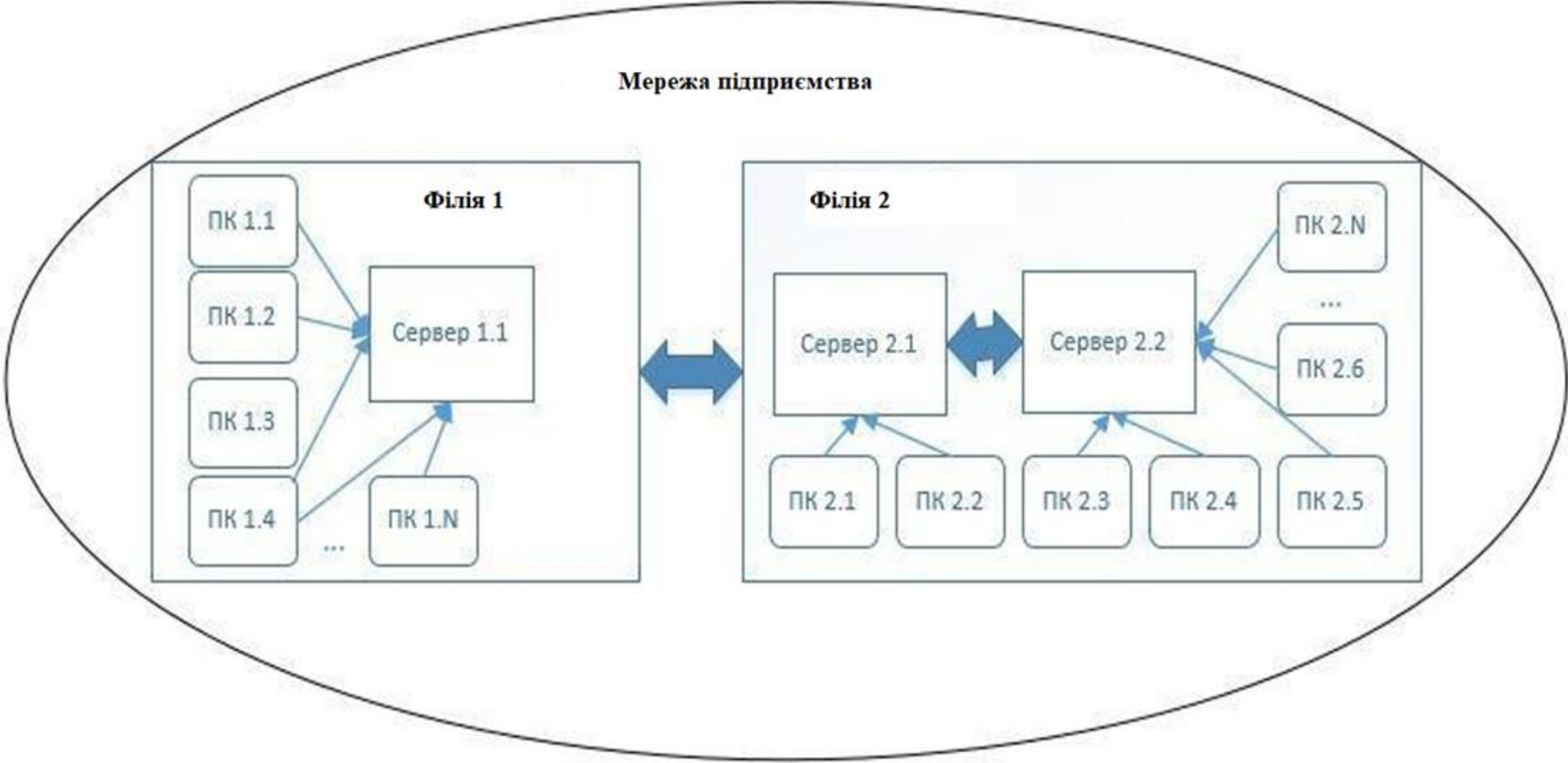
```

        MessageBox.Show("Error: Не вдалося прочитати файл з
        диска. Оригінальна помилка: " + ex.Message);
    }
}

private void DeleteButton_Click(object sender, EventArgs e)
{
    спробувати
    {
        рядок src;
        if (ContentBox.SelectedItem != null)
        {
            src = string.Concat(currentDir.Text,
            ContentBox.SelectedItem.ToString());
            deleteItem(src);
            fillContent(currentDir.Text);
        }
    }
    catch (System.Exception ex)
    {
        MessageBox.Show(ex.Message, "Помилка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

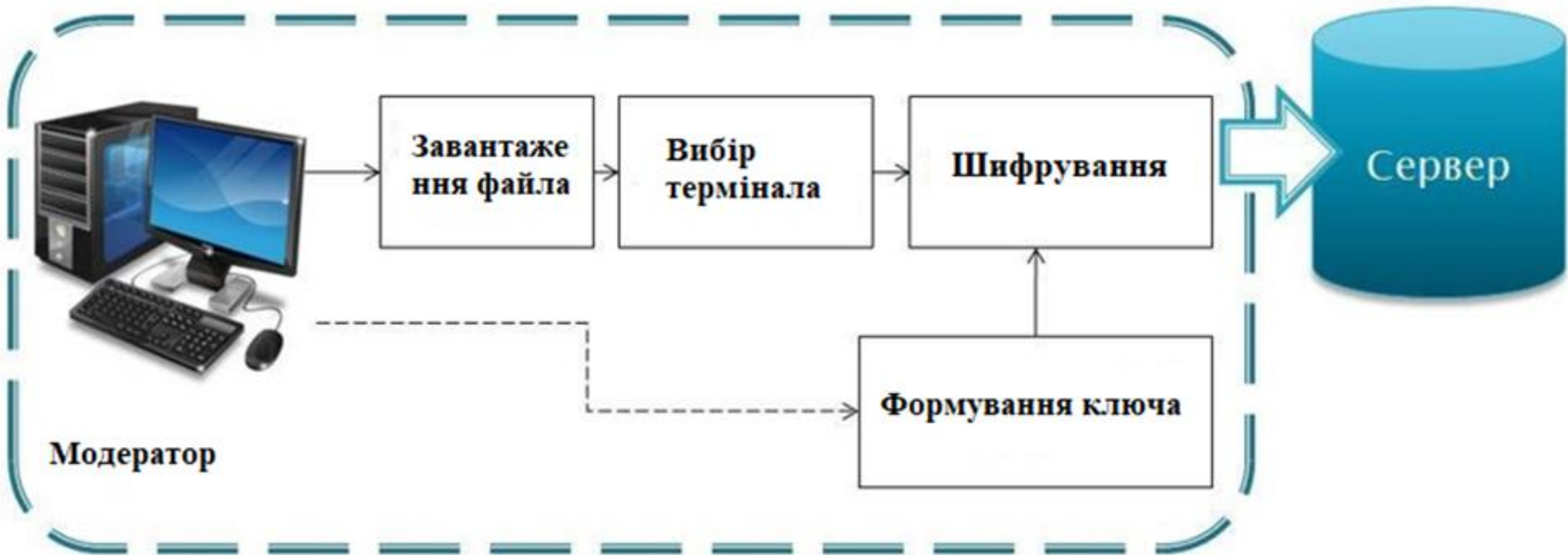
```

Інформаційні ресурси підприємства в корпоративній мережі



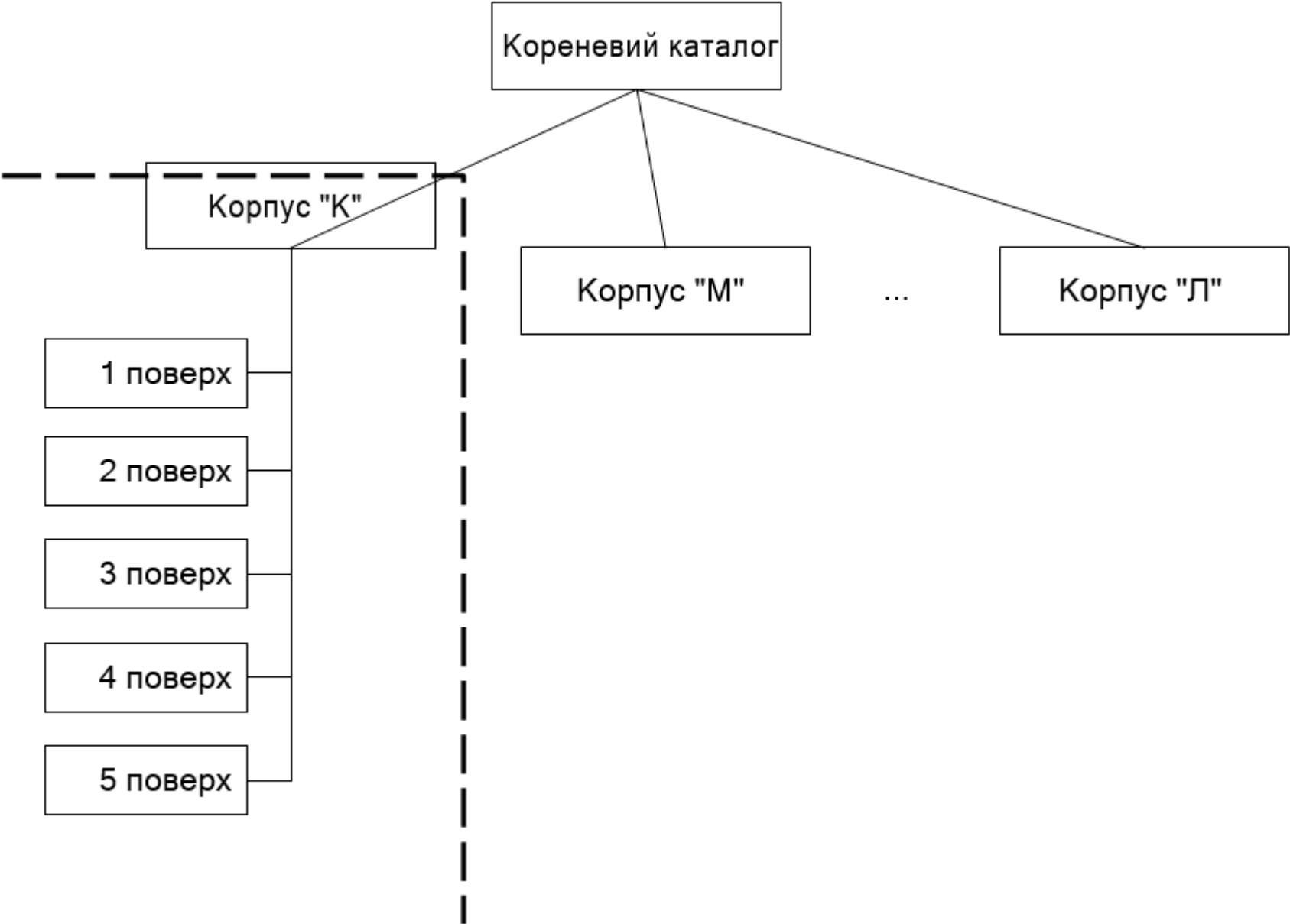
					13.02070849.51921 ПЛ1						
					Інформаційні ресурси підприємства в корпоративній мережі Розробка захищеного хмарного сховища для підприємства «Комтел»	Лист.		Маса		Масшт.	
Зам.	Лист.	№ докум.	Підп.	Дата							
Розроб.		Христенко М.В.									
Перев.		Касьян М.М.									
Т.контр.											
						Лист 1		Листів 1			
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка»					
Затв.		Кудерметов Р.К.				КНТ-519					

Завантаження файлу в сховище



					13.02070849.51921 ПЛ2						
					Завантаження файлу в сховище Розробка захищеного хмарного сховища для підприємства «Комтел»	Лист.		Маса		Масит.	
Зам.	Лист.	№ докум.	Підп.	Дата							
Розроб.		Христенко М.В.									
Перев.		Касьян М.М.									
Т.контр.											
						Лист 1		Листів 1			
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка» КНТ-519					
Затв.		Кудерметов Р.К.									

Ієрархічна структура сховища



					13.02070849.51921 ПЛЗ						
					Ієрархічна структура сховища Розробка захищеного хмарного сховища для підприємства «Комтел»	Лит.		Маса		Масит.	
Зам.	Лист.	№ докум.	Підп.	Дата							
Розроб.		Христенко М.В.									
Перев.		Касьян М.М.									
Т.контр.											
						Лист 1		Листів 1			
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка»					
Затв.		Кудерметов Р.К.				КНТ-519					

Основні етапи роботи системи

File Storage

Введіть логін і пароль

Логин

MyUserName

Пароль

OK

File Storage

Завантажити

Видалити

1\

all\

123.txt

Today08_12.txt

Today11.txt

Today15.txt

Today16_15.txt

					13.02070849.51921 ПЛ4			
					Основні етапи роботи системи Розробка захищеного хмарного сховища для підприємства «Комтел»	Лит.	Маса	Масит.
Зам.	Лист.	№ докум.	Підп.	Дата				
Розроб.		Христенко М.В.						
Перев.		Касьян М.М.						
Т.контр.						Лист 1	Листів 1	
Н.контр.		Щербак Н.В.				НУ «Запорізька політехніка»		
Затв.		Кудерметов Р.К.				КНТ-519		