Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Соловйов Р.А.
«____» _____ 2022 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Соловйову Роману Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Соловйов Р.А.
(прізвище та ініціали)

АНОТАЦІЯ

Соловйов Р.А. Дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Метою розробки є дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Об'єктом дослідження є процес розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Предметом дослідження є методи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Методи дослідження базуються на методах захисту інформації та хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерні науки, розподіл ключів, Cisco SD-WAN, хмарна архітектура

ABSTRACT

Soloviov R.A. Research and software implementation of the key distribution system in the Cisco SD-WAN network based on cloud architecture. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the key distribution system in the Cisco SD-WAN network, based on cloud architecture.

The goal of the development is research and software implementation of a key distribution system in the Cisco SD-WAN network based on cloud architecture.

The object of the study is the key distribution process in the Cisco SD-WAN network based on cloud architecture.

The subject of the research is methods of key distribution in the Cisco SD-WAN network based on cloud architecture.

Research methods are based on methods of information protection and cloud computing, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the key distribution system in the Cisco SD-WAN network, based on cloud architecture.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Builder C++ environment.

Keywords: computer science, key distribution, Cisco SD-WAN, cloud architecture

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГОМ	–	глобальна обчислювальна мережа
ЛОМ	–	локальна обчислювальна мережа
AS	–	сервер автентифікації
DSA	–	алгоритм шифрування
HAKDP	–	Hashed Key Distribution Patterns
HAKDS	–	Hashed Key Distribution Scheme
KDC	–	центр розподілу ключів
KDP	–	Key Distribution Patterns
Kerberos	–	протокол автентифікації
NAS	–	мережевий сервер додатків
OTPS	–	служба одноразових паролів
PKI	–	інфраструктура відкритих ключів
TA	–	довірений центр
TGS	–	Ticket Granting Server
TGT	–	Ticket Granting Ticket
WAN	–	корпоративна мережа

ВСТУП

Актуальність теми. Однією із ключових задач удосконалювання інформаційних комунікацій є задача побудови безпечної комп'ютерної мережі. Інтерес до неї обумовлюється зростаючими об'ємами переданими між учасниками інформаційного обміну конфіденційної інформації, і швидким ростом таких показників інформації, як вартість втрати конфіденційності, вартість схованого порушення цілісності, вартість втрати інформації. Цій проблемі присвячена велика кількість наукових робіт і монографій.

Захищеність комунікацій у безпечній мережі включає забезпечення конфіденційності й цілісності переданої інформації. Ці властивості забезпечуються використовуваними криптографічними системами, успішне функціонування яких припускає використання на приймальні й передавальній сторонах захищеного каналу криптографічних ключів, бінарних наборів достатньої довжини. До 1976 року використовувалися лише симетричні криптосистеми, у яких ключі передавальної й приймаючої сторони повинні бути секретними й практично однаковими, що пов'язане із проблемою доставки ключа від одного учасника іншому або від довіреного центра обом учасникам. З винаходом У. Діффі, М. Хеллманом публічної криптографії один із ключів може бути відкритим і доставлятися його стороні, що використовує, не по закритому, а по автентичному каналі, що вимагає реєстрації цього ключа в центрах сертифікації інфраструктури розподілу відкритих ключів.

Помітимо, що використання криптосистем з відкритим ключем саме по собі не досить для забезпечення захищеності комунікацій у комп'ютерних мережах, оскільки алгоритми криптографії з відкритим ключем через необхідність виконання алгебраїчних операцій в алгебраїчних структурах високих порядків на три порядки повільніше алгоритмів симетричних криптосистем. Тому криптосистеми з відкритим ключем можуть

використовуватися для захисту лише невеликих обсягів інформації й в основному відіграють допоміжну роль, забезпечуючи захист передачі секретних ключів для симетричних криптосистем.

Безпосереднє використання цього способу доставки секретного ключа кожним учасником комп'ютерної мережі приводить до багаторазового (по числу учасників) виконанню дорогих актів сертифікації й використанню ключів, генеруємых учасниками без належного контролю їхньої якості.

Для спрощення процедури формування й доставки секретних ключів у криптографії запропоновані й досліджені різноманітні схеми попереднього розподілу ключів. У них процедура доставки секретного ключа учасникам комп'ютерної мережі виконується у два етапи: кожному учасникові довіреним центром доставляється пакет ключової інформації (у вигляді наборів двійкових слів достатньої довжини), склад якого (можливо, з деякою додатковою відкритою інформацією про ці слова) публікується.

При цьому кожний учасник, знаючи склади пакетів і опубліковані дані, може, використовуючи тільки набори зі свого пакета, обчислити для захищеної комунікації з будь-яким іншим учасником мережі ключ, що не може обчислити ніякий третій учасник.

Розвиваючи цей підхід, криптографи запропонували й схеми більш загального характеру, що дозволяють попередньо розподіляти ключову інформацію для обчислення ключів привілейованих груп учасників, недоступних забороненим групам учасників. Але й цей підхід зустрічає труднощі відносно до великих обчислювальних мереж, оскільки припускає використання єдиного центра генерації й доставки пакетів ключової інформації кожному учасникові мережі. Використання для доставки пакетів криптосистем з відкритим ключем припускає сертифікацію відритих ключів учасниками.

Розглянутий стан проблеми розподілу ключів дозволяє вважати **актуальними** наступні задачі:

– розробка й обґрунтування архітектурних мережевих рішень

нецентралізованого попереднього розподілу ключової інформації в комп'ютерній мережі на основі незалежного попереднього її розподілу в сегментах або доменах мережі;

– розробка й обґрунтування способу реалізації попереднього розподілу ключової інформації в сегментах або доменах обчислювальної мережі на основі використання пропонованої модифікації протоколу Kerberos;

– розробка й обґрунтування нових схем попереднього розподілу ключів;

– розробка програмного забезпечення для обчислення пакетів ключової інформації й принципів побудови системного програмного забезпечення для обчислювальних систем з нецентралізованим попереднім розподілом ключів.

Варто підкреслити відповідність цих напрямків дослідження особливостям мобільних обчислювальних мереж, у яких практично важко реалізовані вимоги сертифікації відкритих ключів і актуальна задача обліку умови розширюваності мережі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– Дослідження системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– Програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Об'єктом дослідження є процес розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Предметом дослідження є методи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Методи дослідження базуються на методах захисту інформації та хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– Розроблено вітчизняний продукт розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система, яка розробляється у ході виконання магістерської роботи, призначена для захисту інформації всередині корпоративної мережі WAN. Особливу увагу приділено такому аспекту захисту інформації, як розподіл ключів, між учасниками інформаційного обміну у цій системі.

Перш ніж розглянути питання розподілу ключів, розглянемо, у якій структурі їх потрібно розподіляти, тобто розглянемо архітектуру WAN.

Корпоративні мережі використовують традиційні технології LAN і WAN. У стандартній корпоративній мережі кілька локальних мереж у комплексі будинків з'єднуються на рівні розподілу або центральному рівні й формують LAN. Ці локальні LAN з'єднуються з іншими географічно розосередженими площадками, формуючи WAN.

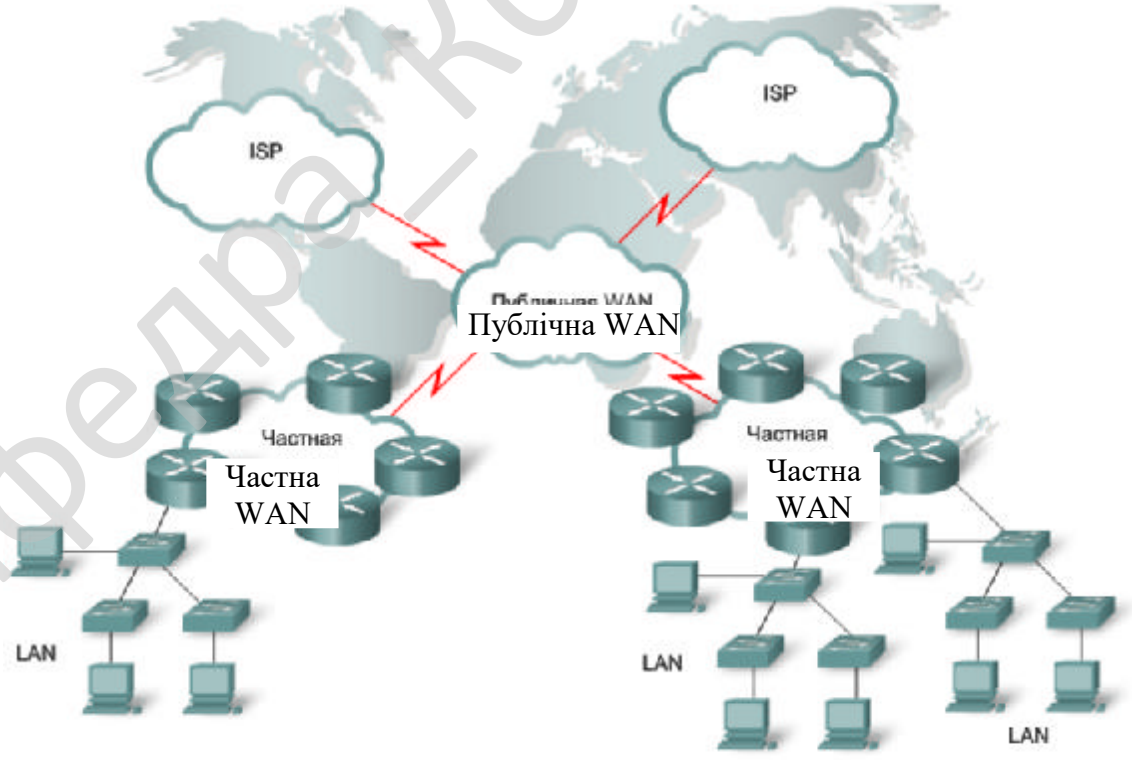

Рисунок 1.1 – Загальна структура корпоративної мережі

LAN є частками й перебувають під контролем однієї особи або організації. Організація встановлює, контролює й обслуговує кабелі й пристрої, що є будівельними блоками LAN.

Деякі WAN є частками, але через високу вартість розробки й обслуговування часток WAN тільки дуже великі організації можуть дозволити собі такі мережі. Більшість компаній купують канали WAN у постачальників послуг (Інтернету). У цьому випадку постачальник послуг Інтернету виконує обслуговування серверної частини системи, тобто внутрішніх мережевих підключень і мережевих послуг між LAN.

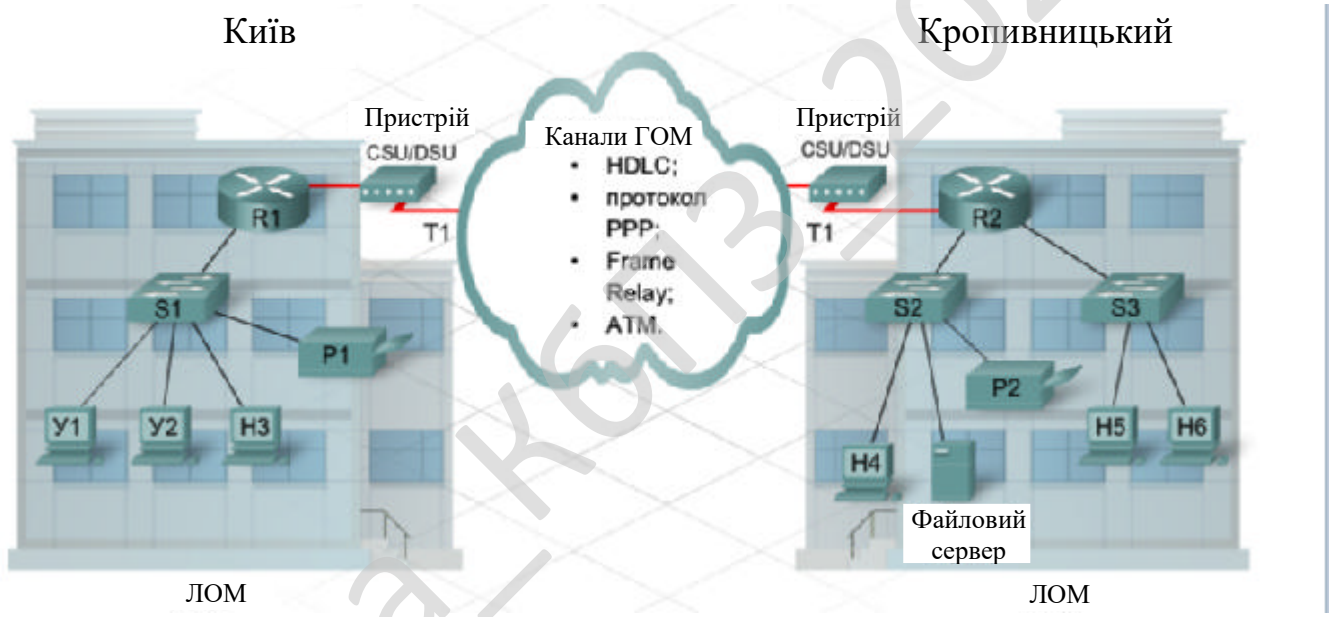

Рисунок 1.2 – Розподілена WAN

З ростом компаній їхнє єдине місце розташування найчастіше розширюється на кілька віддалених об'єктів. Таке розширення вимагає переходу від локальної мережі (LAN) до використання глобальної мережі (WAN).

Якщо організація має значне число площадок по усьому світі, створення каналів і послуг WAN може бути складною задачею. Наприклад, основний постачальник послуг Інтернету організації може не пропонувати обслуговування у всіх населених пунктах і країнах, у яких в організації є офіси. У результаті організації доводиться здобувати послуги в декількох постачальників послуг

Інтернету. Використання декількох постачальників послуг Інтернету часто стає причиною неоднорідної якості надаваних послуг. У багатьох країнах, що розвиваються, розроблювачі мереж можуть зіштовхнутися з розходженнями в доступності пристроїв, послуг WAN і технологій шифрування для безпеки. Для підтримки корпоративної мережі важливо мати єдині стандарти встаткування, конфігурації й послуг.

Усередині локальної мережі всіма кабельними з'єднаннями, пристроями й службами адміністратор мережі управляє на місці. Хоча деякі великі компанії самостійно обслуговують мережі WAN, більшість організацій отримують служби глобальної мережі в постачальників послуг WAN.

За користування мережевими ресурсами постачальники послуг стягують плату. Постачальники послуг Інтернету дають можливість користувачам розподіляти ресурси між віддаленими об'єктами без дорогих витрат на будівництво й обслуговування власної мережі.

Керування мережевими ресурсами не є єдиним розходженням між локальною (LAN) і глобальної (WAN) мережами. Вони розрізняються також за технологіями. Найпоширеніша технологія локальної мережі (LAN) – Ethernet. Технології мережі WAN полягають у послідовній передачі даних. Послідовна передача даних дозволяє підтримувати надійні з'єднання на далекі відстані з меншою швидкодією, ніж локальна мережа (LAN).

Розглянемо функцій WAN та LAN.

Функції LAN:

– організація відповідає за установку інфраструктури й керування нею;

– Ethernet – найпоширеніша з використовуваних технологій;

– вона призначена для рівнів доступу й розподілу;

– LAN з'єднує користувачів і надає підтримку локалізованих додатків і серверних ферм;

– з'єднані пристрої, як правило, перебувають у тій же локальній області, наприклад у будинку або комплексі будинків.

Функції WAN:

– з'єднання площадок, які, як правило, перебувають на значній відстані одна від одної;

– підключення до WAN вимагає пристрою, що перетворить дані у форму, прийнятну для мережі постачальника послуг, наприклад модему або пристрою CSU/DSU;

– послуги надаються постачальником послуг Інтернету. Типи послуг WAN: T1/T3, E1/E3, DSL, кабельне з'єднання, Frame Relay і ATM;

– відповідальність за установку інфраструктури й керування нею несе постачальник послуг Інтернету;

– прикордонні пристрої перетворять інкапсуляцію Ethernet у послідовну інкапсуляцію WAN.

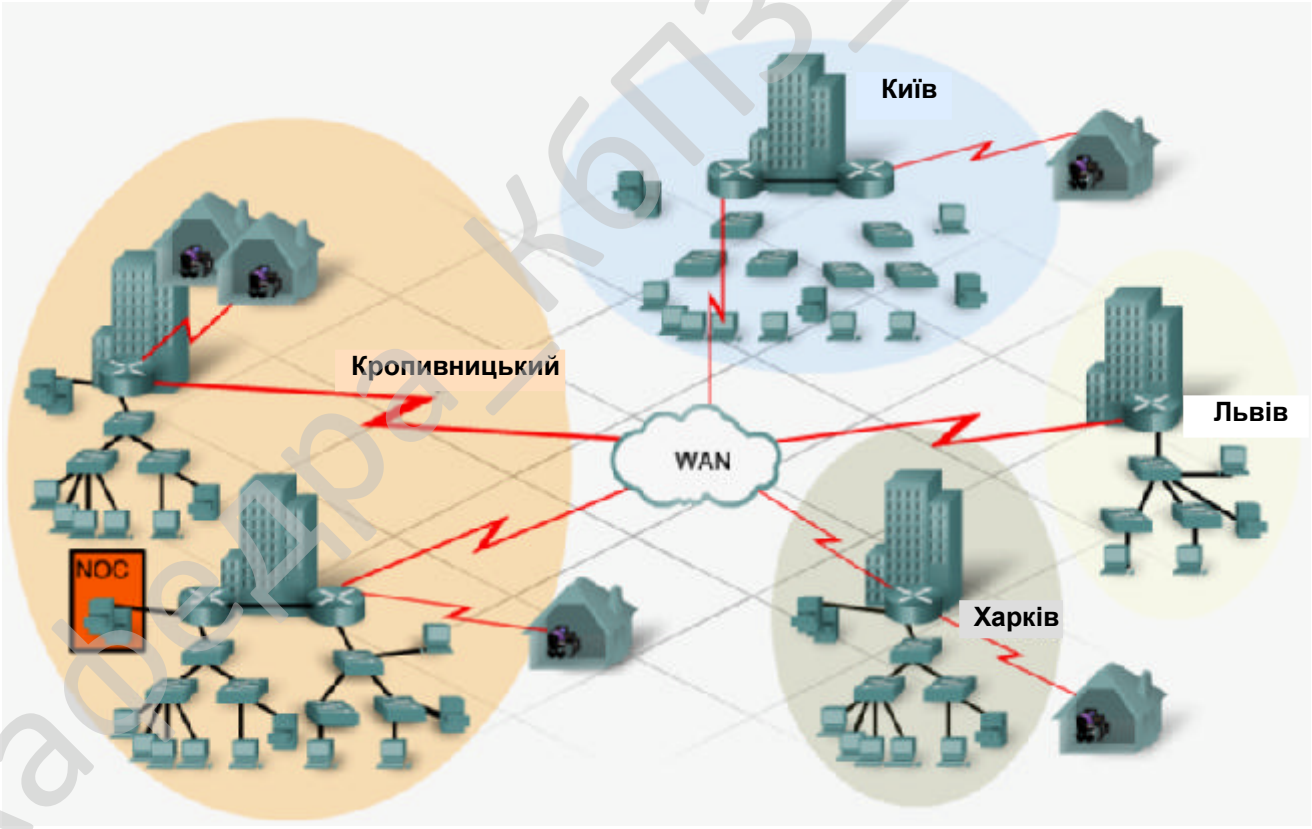

Рисунок 1.3 – Схема розподіленої WAN

1.2 Область застосування

У мережах WAN існує необхідність у захисті інформації, яка циркулює всередині корпорації. Однією з найважливіших задач забезпечення інформаційної безпеки є захист потоків корпоративних даних, переданих по каналах загального користування, у тому числі й через Internet. Перспективним методом надійного захисту інформації є метод кодування даних.

Для рішення цієї задачі необхідно здійснити кодування інформації на виході з локальної мережі й декодування вступників у неї даних. Ці функції реалізуються спеціальними програмними або програмно-апаратними засобами. Якщо захист сегмента корпоративної мережі вже забезпечена міжмережевим екраном, природно покласти на нього також виконання функцій кодування й декодування.

Для реалізації можливостей кодування/декодування повинне бути виконане попередній (початковий) розподіл ключів. Сучасні технології пропонують для цього цілий ряд методів. Після розподілу ключів з'являється можливість здійснення процесу вироблення спільних секретних ключів, що обслуговують сеанс спілкування абонентів.

У результаті кодування весь обмін даними між територіально-віддаленими локальними мережами є захищеним і для користувачів виглядає як обмін усередині однієї локальної мережі, при цьому від користувачів не потрібне застосування яких-небудь додаткових захисних засобів.

Інфраструктура безпеки для поширення відкритих ключів, керування електронними сертифікатами й ключами користувачів одержала назву інфраструктури відкритих ключів – Public Key Infrastructure (PKI).

Термін "PKI" є похідним від назви базової технології – криптографії з відкритими ключами, що володіє унікальними властивостями й основою, що є, для реалізації функцій безпеки в розподілені системах. інфраструктура відкритих ключів реалізується не заради її самої, а для підтримки безпеки інших додатків.

Існують, безумовно, і інші механізми безпеки, які не використовують криптографію відкритих ключів, і вони менш складні, чим PKI. Однак PKI не тільки пропонує найбільш комплексне рішення, але й знімає гостроту багатьох проблем, властивих більш традиційним механізмам безпеки.

Корпоративні системи звичайно працюють із більшою кількістю користувачів, причому кожний користувач взаємодіє із системою по-своєму. Автентифікація логічно стає першим кроком при взаємодії людини із системою. Механізми автентифікації розвивалися із часом, опираючись на властивості своїх попередників. Автентифікація на базі PKI – це логічний крок у цій еволюції, що забезпечує вдосконалювання таких характеристик, як застосовність, життєздатність, масштабованість і безпека.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Розглянемо відомі методи й протоколи розподілу ключів і протокол Kerberos у системі розподілу ключів. Дамо визначення схем попереднього розподілу ключів і попереднього розподіл системних ключів. Розглянемо їхні основні властивості, класи й приклади схем.

Автентифікація в Windows

Windows підтримує кілька протоколів, які дозволяють переконатися в тому, що вхідний у систему користувач дійсно має тут свій обліковий запис. Серед них – протоколи автентифікації віддалених підключень і протоколи автентифікації користувачів, що входять у мережу через Інтернет. Однак усередині доменів Windows для перевірки користувальницьких даних передбачено два методи:

– **Kerberos.** Протокол Kerberos є стандартним засобом автентифікації мережевих користувачів у домені Active Directory на всіх комп'ютерах з операційною системою Windows.

– **Windows NT LAN Manager (NTLM).** Протокол NTLM застосовувався як стандартний засіб мережевої автентифікації в операційній системі Windows. У середовищі Windows він використовується для автентифікації серверів і доменів Windows. Крім того, NTLM застосовується для локальної автентифікації на автономних комп'ютерах, що працюють під керуванням Windows.

Переваги автентифікації по протоколу Kerberos

Протокол Kerberos вигідно відрізняється від NTLM більшою гнучкістю й ефективністю використання. Забезпечує він і підвищений рівень безпеки.

Наведемо ряд переваг, які дає перехід на Kerberos.

– Більш ефективна автентифікація на серверах. При автентифікації по протоколу NTLM серверу додатків доводиться підключатися до контролера домена при перевірці кожного клієнта. З Kerberos така необхідність відпадає – тут автентифікація виробляється за рахунок перевірки посвідчення, представленого клієнтом. Індивідуальне посвідчення клієнт одержує від контролера один раз, після чого може неодноразово використовувати його протягом усього сеансу роботи в мережі.

– Взаємна автентифікація. Протокол NTLM дозволяє серверу ідентифікувати своїх клієнтів, однак не передбачає верифікації сервера ні клієнтами, ні іншими серверами. Цей протокол розроблявся для мереж, у яких всі сервери вважаються легітимними. На відміну від його, Kerberos такого допущення не робить, тому перевіряє обох учасників мережевого підключення, кожний з яких у результаті може точно довідатися, з ким підтримує зв'язок.

– Делегована автентифікація. Коли клієнт мережі Windows звертається до ресурсів, служби операційної системи, насамперед, роблять його ідентифікацію. У багатьох випадках для виконання цієї операції службі досить інформації на локальному комп'ютері. Як NTLM, так і Kerberos забезпечують всі дані, необхідні для ідентифікації користувача на місці, однак іноді їх буває недостатньо. Деякі розподілені додатки вимагають, щоб при підключенні до серверних служб на інших комп'ютерах ідентифікація клієнта вироблялася локально службою самого цього клієнта. Вирішити проблему допомагає Kerberos, де передбачений спеціальний механізм представницьких білетів, що дозволяє на місці ідентифікувати клієнта при його підключенні до інших систем. У протоколі NTLM така можливість відсутня.

– Спрощене керування довірчими відносинами. Одне з важливих достоїнств взаємної автентифікації по протоколу Kerberos полягає в тому, що довірчі відносини між доменами Windows за замовчуванням є двосторонніми й транзитивними. Завдяки цьому в мережах із множиною доменів не прийдеться

встановлювати багато явних довірчих відносин. Замість цього всі домени великої мережі можна звести в дерево транзитивних відносин взаємної довіри. Посвідчення, видане системою безпеки для будь-якого домена, може прийматися у всіх галузях дерева. Якщо ж мережа містить кілька дерев, то посвідчення кожного з них буде прийматися по всьому «лісі».

– Сумісність. В основу своєї реалізації протоколу Kerberos покладені стандартні специфікації, рекомендовані групою IETF..

Стандарти автентифікації по протоколу Kerberos

Реалізація протоколу в Windows виконана в строгій відповідності з вимогами, викладеними в документі RFC 1510.

Крім того, при її розробці були використані механізм і формат передачі контекстів безпеки в повідомленнях Kerberos, описані в специфікаціях Інтернету з документа RFC 1964.

Розширення протоколу Kerberos

В Windows знайшли застосування розширення протоколу Kerberos, що спрощують початкову автентифікацію клієнтів. Зазвичай для цієї мети використовуються секретні ключі, якими повинні заздалегідь обмінятися між собою учасники сеансу, але тепер таку процедуру можна провести за допомогою відкритих ключів. Завдяки цьому з'явилася можливість інтерактивної реєстрації користувача за допомогою мікропроцесорних карток. В основу розширень, що забезпечують автентифікацію з відкритим ключем, покладена специфікація PKINIT.

Огляд протоколу Kerberos

Протокол автентифікації Kerberos пропонує механізм взаємної ідентифікації клієнта й сервера (або двох серверів) перед установленням зв'язку між ними.

У протоколі враховано, що початковий обмін інформацією між клієнтами й серверами відбувається у відкритому середовищі, а пакети, передані по каналах зв'язку, можуть бути перехоплені й модифіковані.

Інакше кажучи, протокол призначений для роботи в середовищі, що дуже нагадує сьогоднішній Інтернет. Тут зловмисник легко може імітувати запити клієнта або сервера, перехоплювати зв'язок між легітимними клієнтами й серверами, навіть спотворювати передану інформацію.

Основні концепції

Протокол Kerberos активно використовує технології автентифікації, що опираються на «секрети для двох». Основна концепція досить проста: якщо є секрет, відомий тільки двом, будь-який з його хоронителів може легко впевнитися, що має справу саме зі своїм напарником. Для цього йому досить яким-небудь способом перевірити, чи знає співрозмовник їхній загальний секрет.

Розглянемо це на простому прикладі. Припустимо, Сторона А часто посилає повідомлення Стороні Б, що використовує інформацію, яка втримується в них, тільки тоді, коли повністю впевнена, що вона надійшла саме від Сторони А.

Щоб ніхто інший не зміг підробити листа, вони домовилися про пароль, що пообіцяли нікому більше не говорити. Якщо з повідомлення можна буде укласти, відправник, що знає цей пароль, Сторона Б може точно сказати: воно прийшло від Сторони А.

Тепер Стороні Б и Стороні А залишається тільки вирішити, *яким чином* показати своє знання пароля. Можна, скажемо, просто включити його в текст повідомлення, наприклад, після підпису. Це було б дуже просто, зручно й надійно, якби Сторона А и Сторона Б були впевнені, що ніхто інший не читає їхніх повідомлень. Сторона А не може просто назвати пароль у тексті листа. Щоб секрет залишався секретом, про нього не можна говорити відкрито, потрібно знайти спосіб тільки сповістити співрозмовникові, що він тобі відомий.

У протоколі Kerberos ця проблема вирішується засобами криптографії із секретним ключем. Замість того щоб повідомляти один одному пароль, учасники сеансу обмінюються криптографічним ключем, знання якого підтверджує особистість співрозмовника. Один з учасників використовує такий ключ для

шифрування блоку інформації, а іншої з його допомогою витягає цю інформацію.

Автентифікатори

Простий протокол автентифікації із секретним ключем вступає в дію, коли хто-небудь стукається в мережеві двері й просить впустити його. Щоб довести своє право на вхід, користувач пред'являє **автентифікатор** (authenticator) у вигляді блоку інформації, зашифрованого за допомогою секретного ключа. Зміст цього блоку повинен мінятися при кожному наступному сеансі, у противному випадку зловмисник цілком може проникнути в систему, скориставшись перехопленим повідомленням. Одержавши автентифікатор, воротар розшифровує його й перевіряє отриману інформацію, щоб переконатися в успішності розшифрування. Якщо все пройшло нормально, страж може бути впевнений, що людині, яка пред'явила автентифікатор, відомий секретний ключ. Але ж цей ключ знають усього двоє, причому один з них – сам воротар. Таким чином, він робить висновок, що прибулець дійсно та людина, за яку себе видає.

Але може трапитися й так, що суб'єкт, що постукався у двері, захоче провести взаємну автентифікацію. У цьому випадку використовується той же самий протокол, але у зворотному порядку й з деякими змінами. Тепер воротар витягає з вихідного автентифікатора частину інформації, а потім шифрує її, перетворюючи в новий автентифікатор, і в такому виді повертає прибульцеві. Той, у свою чергу, розшифровує отриману інформацію й порівнює її з вихідною. Якщо все збігається, прибулець може бути впевнений: раз воротар розшифрував оригінал, виходить, він знає секретний ключ.

А тепер повернемося до нашого приклада. Припустимо, Сторона А и Сторона Б домовилися: перед тим, як пересилати інформацію між своїми комп'ютерами, вони за допомогою відомого тільки їм двом секретного ключа, будуть перевіряти, хто перебуває на іншому кінці лінії. У ситуації, коли Сторона А відіграє роль обережного гостя, а Сторона Б – пильного воротаря, це буде виглядати так.

1. Сторона А посилає Стороні Б повідомлення, що містить її ім'я у

відкритому виді й автентифікатор, зашифрований за допомогою секретного ключа, відомого тільки їм двом. У протоколі автентифікації таке повідомлення являє собою структуру даних із двома полями. Перше поле містить інформацію про Сторону А – її ім'я. У другому полі вказується поточний час на робочій станції Сторони А.

2. Сторона Б одержує повідомлення й бачить, що воно прийшло від когось, хто називає себе Стороною А. Він відразу ж дістає ключ, яким вони зі Стороною А домовилися шифрувати автентифікатор, і, розшифрувавши друге поле, довідується час відправлення повідомлення.

Задача Сторони Б набагато спрощується, якщо її комп'ютер працює синхронно з комп'ютером Сторони А, тому давайте припустимо, що обоє вони звіряють свої годинники з мережевим часом, завдяки чому ті йдуть практично однаково. Допустимо, годинники на комп'ютерах Сторони А и Сторони Б ніколи не розходяться більше, ніж на п'ять хвилин. У цьому випадку Сторона Б може зрівняти витягнуте із другого поля автентифікатора значення з поточним часом на своїх годинниках. Якщо розходження складе більше п'яти хвилин, комп'ютер автоматично відмовиться визнавати дійсність автентифікатора.

Якщо ж час виявляється в межах припустимого відхилення, можна з великою часткою впевненості припустити, що автентифікатор надійшов саме від Сторони А. Але Стороні Б цього мало, їй потрібна повна впевненість. Адже, міркує вона, може бути й так: хтось перехопив попередню спробу Сторони А зв'язатися із мною й тепер намагається скористатися її автентифікатором. Втім, якщо на комп'ютері збереглися записи про час автентифікаторов, що надійшли від Сторони А за останні п'ять хвилин, Сторона Б може знайти останній і відмовитися від всіх інших повідомлень, відправлених одночасно з ним або раніше. Але якщо друге поле свідчить, що повідомлення пішло в мережу вже після відправлення останнього автентифікатора Сторони А, то і його автором цілком могла бути Сторона А.

3. Сторона Б шифрує час із повідомлення Сторони А за допомогою

їхнього загального ключа й включає його у власне повідомлення, що направляє Стороні А.

Зверніть увагу, що Сторона Б повертає Стороні А не всю інформацію з її автентифікатора, а тільки час. Якби вона відправила все відразу, у Сторони А закралася б підозра, що хтось, вирішивши прикинутися Стороною Б, просто скопіював автентифікатор з її вихідного повідомлення й без яких-небудь змін відправив його назад. Але в отриманому листі втримується тільки частина інформації, а це значить, що одержувач вихідного автентифікатора зміг розшифрувати його й обробити інформацію, що втримується там. А час він вибрав тому, що це поле є унікальним ідентифікатором повідомлення Сторони А.

Сторона А одержує відповідь Сторони Б, розшифровує її, а потім порівнює отриманий результат згодом, що було зазначено у вихідному автентифікаторі. Якщо ці дані збігаються, вона може бути впевнена, що її автентифікатор дійшов до того, хто знає їх зі Стороною Б секретний ключ. Ця людина змогла розшифрувати повідомлення й витягти з нього інформацію про час. Оскільки ні вона, ні Сторона Б нікому свій ключ не передавали, Сторона А робить висновок, що саме Сторона Б одержав її автентифікатор і відповів на нього.

Керування ключами

При використанні простих протоколів, на зразок описаного вище, виникає одна дуже важлива проблема. У випадку зі Стороною А и Стороною Б ми ні слова не сказали про те, як і де вони домовлялися про секретний ключ для своєї переписки. Звичайно, люди можуть просто зустрітися в парку й обговорити всі деталі, але адже в мережевих переговорах беруть участь машини. Якщо під Стороною А розуміти клієнтську програму, установлену на робочій станції, а під Стороною Б – службу на мережевому сервері, то зустрітися вони ніяк не можуть. Проблема ще більше ускладнюється в тих випадках, коли Стороні А – клієнтові – потрібно посилати повідомлення на кілька серверів, адже для кожного з них прийдеться обзавестися окремим ключем. Та й службі на Стороні Б буде потрібно

стільки секретних ключів, скільки в неї клієнтів. Але якщо кожному клієнтові для підтримки зв'язку з кожною службою потрібен індивідуальний ключ, і такий же ключ потрібний кожній службі для кожного клієнта, то проблема обміну ключами дуже швидко здобуває граничну гостроту. Необхідність зберігання й захисту такої множини ключів на величезній кількості комп'ютерів створює неймовірний ризик для всієї системи безпеки.

Сама назва протоколу Kerberos говорить про те, як тут вирішена проблема керування ключами. Кербер (або *Цербер*) – персонаж класичної грецької міфології. Цей лютий пес з трьома головами, по повір'ях греків, охороняє врата підземного царства мертвих. Трьом головам Кербера в протоколі Kerberos відповідають три учасники безпечного зв'язку: клієнт, сервер і довірений посередник між ними. Роль посередника тут грає так званий центр розподілу ключів **Key Distribution Center** або, скорочено, KDC.

KDC являє собою службу, що працює на фізично захищеному сервері. Вона веде базу даних з інформацією про облікові записи всіх головних абонентів безпеки (security principal) своєї області (області Kerberos у мережах Windows відповідає домен, тому надалі ми будемо застосовувати саме цей звичний термін). Разом з інформацією про кожного абонента безпеки в базі даних KDC зберігається криптографічний ключ, відомий тільки цьому абонентові й службі KDC. Даний ключ, що називають **довгостроковим**, використовується для зв'язку користувача системи безпеки із центром розподілу ключів. У більшості практичних реалізацій протоколу Kerberos довгострокові ключі створюються на основі пароля користувача.

Коли клієнтові потрібно звернутися до сервера, він, насамперед, направляє запит у центр KDC, що у відповідь направляє кожному учасникові майбутнього сеансу копії унікального **сеансового ключа** (session key), що діють протягом короткого часу. Призначення цих ключів – проведення автентифікації клієнта й сервера. Копія сеансового ключа, що пересилається на сервер, шифрується за допомогою довгострокового ключа цього сервера, а та, що направляється

клієнтові – довгострокового ключа даного клієнта.

Теоретично, для виконання функцій довіреного посередника центру KDC досить направити сеансові ключі безпосередньо абонентам безпеки, як показано вище. Однак на практиці реалізувати таку схему надзвичайно складно. Насамперед, серверу довелося б зберігати свою копію сеансового ключа в пам'яті доти, поки клієнт не зв'яжеться з ним. Але ж сервер обслуговує не одного клієнта, тому йому потрібно зберігати паролі всіх клієнтів, які можуть зажадати його уваги. У таких умовах керування ключами вимагає значної витрати серверних ресурсів, що обмежує масштабованість системи. Не можна забувати й про превратності мережевого трафика. Вони можуть привести до того, що запит від клієнта, що вже одержав сеансовий пароль, надійде на сервер раніше, ніж повідомлення KDC із цим паролем. У результаті серверу прийдеться почекати з відповіддю доти, поки він не одержить свою копію сеансового пароля. Тобто, потрібно буде зберегти стани сервера, а це накладає на його ресурси ще один тяжкий тягар. Тому на практиці застосовується інша схема керування паролями, що робить протокол Kerberos набагато більше ефективним. Її опис приводиться нижче.

Сеансові білети

У відповідь на запит клієнта, що має намір підключитися до сервера, служба KDC направляє обидві копії сеансового ключа клієнтові. Повідомлення, призначене клієнтові, шифрується за допомогою довгострокового ключа клієнта, а сеансовий ключ для сервера разом з інформацією про клієнта вкладається в блок даних, що одержав назву **сеансового білета** (session ticket). Потім сеансовий білет цілком шифрується за допомогою довгострокового ключа сервера, що знають тільки служба KDC і даний сервер. Після цього вся відповідальність за обробку білета, що несе в собі зашифрований сеансовий ключ, покладається на клієнта, що повинен доставити його на сервер.

Зверніть увагу, що в цьому випадку функції служби KDC обмежуються видачею білета. Їй більше не потрібно стежити за тим, чи всі відправлені

повідомлення доставлені відповідним адресатам. Навіть якщо яке-небудь із них потрапить не туди, – нічого страшного не трапиться. Розшифрувати клієнтську копію сеансового ключа може тільки той, хто знає секретний довгостроковий ключ даного клієнта, а щоб прочитати вміст сеансового білета, потрібний довгостроковий секретний ключ сервера.

Одержавши відповідь KDC, клієнт витягає з нього сеансовий білет і свою копію сеансового ключа, які поміщає в безпечне сховище (воно розташовується не на диску, а в оперативній пам'яті). Коли виникає необхідність зв'язатися із сервером, клієнт посилає йому повідомлення, що складається із білета, що як і раніше, зашифрований із застосуванням довгострокового ключа цього сервера, і власного автентифікатора, зашифрованого за допомогою сеансового ключа. Цей білет у комбінації з автентифікатором саме й становить посвідчення, по якому сервер визначає «особистість» клієнта.

Сервер, одержавши «посвідчення особи» клієнта, за допомогою свого секретного ключа розшифровує сеансовий білет і витягає з нього сеансовий ключ, що потім використовує для розшифрування автентифікатора клієнта. Якщо все проходить нормально, робиться висновок, що посвідчення клієнта видане довіреним посередником, тобто, службою KDC.

Клієнт може зажадати у сервера проведення взаємної автентифікації. У цьому випадку сервер за допомогою своєї копії сеансового ключа шифрує мітку часу з автентифікатора клієнта й у такому виді пересилає її клієнтові в якості власного автентифікатора.

Одне з достоїнств застосування сеансових білетів полягає в тому, що серверу не потрібно зберігати сеансові ключі для зв'язку із клієнтами. Вони зберігаються в кеш-пам'яті посвідчень (credentials cache) клієнта, що направляє білет на сервер щораз, коли хоче зв'язатися з ним. Сервер, зі своєї сторони, одержавши від клієнта білет, розшифровує його й витягає сеансовий ключ. Коли потреба в цьому ключі зникає, сервер може просто стерти його зі своєї пам'яті.

Такий метод дає й ще одну перевагу: у клієнта зникає необхідність

звертатися до центра KDC перед кожним сеансом зв'язку з конкретним сервером. Сеансові білети можна використовувати багаторазово. На випадок же їхнього розкрадання встановлюється строк придатності білета, що KDC указує в самій структурі даних. Цей час визначається політикою Kerberos для конкретного домена. Звичайно строк придатності білетів не перевищує восьми годин, тобто, стандартної тривалості одного сеансу роботи в мережі. Коли користувач відключається від його, кеш-пам'ять посвідчень обнуляется, і всі сеансові білети разом із сеансовими ключами знищуються.

Білети на видачу білетів

Як ми вже відзначали, довгостроковий ключ користувача створюється на основі його пароля. Щоб краще зрозуміти це, повернемося до нашого приклада. Коли Сторона А проходить реєстрацію, клієнт Kerberos, установлений на її робочій станції, пропускає зазначений користувачем пароль через функцію гешування. У результаті формується криптографічний ключ.

У центрі KDC довгострокові ключі Сторони А зберігаються в базі даних з обліковими записами користувачів. Одержавши запит від клієнта Kerberos, установленого на робочій станції Сторони А, KDC звертається у свою базу даних, знаходить у ній обліковий запис потрібного користувача й витягає з відповідного її поля довгостроковий ключ Сторони А.

Такий процес – обчислення однієї копії ключа по паролю й добування іншої його копії з бази даних – виконується всього лише один раз за сеанс, коли користувач входить у мережу вперше. Відразу ж після одержання користувальницького пароля й обчислення довгострокового ключа клієнт Kerberos робочої станції запитує сеансовий білет і сеансовий ключ, які використовуються у всіх наступних транзакціях з KDC протягом поточного сеансу роботи в мережі.

На запит користувача KDC відповідає спеціальним сеансовим білетом для самого себе, так званий **білет на видачу білетів** (ticket-granting ticket), або білет TGT. Як і звичайний сеансовий білет, TGT містить копію сеансового ключа для

зв'язку служби (у цьому випадку – KDC) із клієнтом. У повідомлення із білетом TGT також включається копія сеансового ключа, за допомогою якої клієнт може зв'язатися з KDC. Білет TGT шифрується за допомогою довгострокового ключа служби KDC, а клієнтська копія сеансового ключа – за допомогою довгострокового ключа користувача.

Одержавши відповідь служби KDC на свій первісний запит, клієнт розшифровує свою копію сеансового ключа, використовуючи для цього копію довгострокового ключа користувача зі своєї кеш-пам'яті. Після цього довгостроковий ключ, отриманий з користувальницького пароля, можна видалити з пам'яті, оскільки він більше не знадобиться: весь наступний зв'язок з KDC буде шифруватися за допомогою сеансового ключа. Як і всі інші сеансові ключі, він має тимчасовий характер і дійсний до закінчення терміну дії білета TGT, або до виходу користувача із системи. Із цієї причини такий ключ називають **сеансовим ключем реєстрації** (logon session key).

З погляду клієнта, білет TGT майже нініж не відрізняється від звичайного. Перед підключенням до будь-якої служби, клієнт, насамперед, звертається в кеш-пам'ять посвідчень і дістає звідти сеансовий білет для цієї служби. Якщо його немає, він починає шукати в цій же кеш-пам'яті білет TGT. Знайшовши його, клієнт витягає звідти ж відповідний сеансовий ключ реєстрації й готовить із його допомогою автентифікатор, що разом з TGT висилає в KDC. Одночасно туди направляється запит на сеансовий білет для необхідної служби. Інакше кажучи, організація безпечного доступу до KDC нічого не відрізняється від організації такого доступу до будь-якої іншої служби домена – вона вимагає сеансового ключа, автентифікатора й білета (у цьому випадку TGT).

Із точки ж зору служби KDC, білети TGT дозволяють прискорити обробку запитів на одержання білетів, заощадивши декілька наносекунд на пересиланні кожного з них. Центр розподілу ключів KDC звертається до довгострокового ключа користувача тільки один раз, коли надає клієнтові первісний білет TGT. У всіх наступних транзакціях із цим клієнтом центр KDC розшифровує білети TGT

за допомогою власного довгострокового ключа й витягає з нього сеансовий ключ реєстрації, що використовує для перевірки дійсності автентифікатора клієнта.

Автентифікація за межами домена

Функції центра KDC можна розділити на дві категорії:

– службу автентифікації, у задачу якої входить генерація білетів TGT;

– службу видачі білетів, що створює сеансові білети.

Такий поділ праці дозволяє застосовувати протокол Kerberos і за межами його «рідного» домена. Клієнт, що одержав білет TGT зі служби автентифікації одного домена, може скористатися ним для одержання сеансових білетів у службах видачі білетів інших доменів.

Процес переадресації запиту трохи ускладнюється в мережах, де розгорнуто більше двох доменів. Теоретично служба KDC кожного домена може встановити безпосередній зв'язок з аналогічними службами всіх доменів мережі, домовившись із кожною з них про індивідуальний міждомений ключ.

Однак на практиці кількість і складність подібних взаємозв'язків дуже швидко зростають настільки, що стають просто некерованими, особливо в складних мережах. Протокол Kerberos вирішує цю проблему, роблячи непотрібними прямі зв'язки між доменами. Клієнт одного домена може одержати білет на доступ до сервера іншого домена через один або кілька проміжних серверів, кожний з яких видасть йому білет переадресації.

Підпротоколи

Протокол Kerberos містить у собі три підпротокола. Перший з них використовується службою KDC для передачі клієнтові сеансового ключа реєстрації й білета TGT. Він називається Authentication Service Exchange (обмін зі службою автентифікації) або, скорочено AS Exchange.

Другий підпротокол за назвою Ticket-Granting Service Exchange (обмін зі службою видачі білетів) або TGS Exchange служить для розсилання службових сеансових ключів і сеансових ключів самої служби KDC.

Третій підпротокол, Client/Server Exchange (клієнт-серверний обмін) або

CS Exchange, використовується клієнтом для пересилання сеансового білета доступу до служб.

Більш докладно підпротоколи будуть наведені нижче.

Білети

Вище ми говорили про білети лише загалом. Тепер прийшов час більш докладно розглянути, що ж це таке, як розраховується строк придатності білета, і яка його частина стає відомою клієнтові. Всі ці деталі важливі для розробки політики Kerberos, тому до них потрібно придивитися якнайближче.

У рамках даної роботи досить перелічити поля білета й описати інформацію, що втримується в них. Більш докладно структура білета й різних повідомлень Kerberos описана в документі RFC 1510.

Вміст поля *flags* адресується побітно. Включення й вимикання прапорів тут виробляється зміною значення (0 або 1) відповідного біта. Довжина поля – 32 розряду, однак для адміністратора Kerberos інтерес представляють тільки 9 прапорів білета.

Призначення протоколу Kerberos у загальному значенні ширше досліджуваної в магістерській роботі проблеми розподілу ключів: він надає собою розроблений для мереж TCP/IP протокол перевірки дійсності з довіреною третьою стороною. Використання цього стандарту створює надійну основу для взаємодії між різними платформами й при цьому значно підвищується безпека мережевої автентифікації.

Служба Kerberos, що працює в мережі, діє як довірений посередник, забезпечує перевірку дійсності мережевих об'єктів. Об'єктами мережевої взаємодії в моделі Kerberos є клієнти й сервери. Kerberos зберігає базу даних клієнтів і їхніх секретних ключів.

Мережеві служби, що вимагають перевірки дійсності, і клієнти, які хочуть використовувати ці служби, реєструють в Kerberos свої секретні ключі. Так як Kerberos знає всі секретні ключі, він може створювати повідомлення, що переконують один об'єкт у дійсності іншого.

Kerberos також створює унікальні сеансові ключі, які видаються клієнтові й серверу (або двом клієнтам).

Сеансовий ключ використовується для шифрування/розшифрування повідомлення, яким обмінюються дві сторони, і знищуються після закінчення сеансу. Таким чином, протоколом Kerberos, зокрема, вирішується задача розподілу ключів для комунікації між двома абонентами мережі.

Таблиця 2.1 – Поля білета

Назва поля	Опис
Перші три поля білета не шифруються. інформація, Що втримується тут, пересилається відкритим текстом, що дозволяє клієнтові використовувати неї для керування білетами, що зберігаються в кеш-пам'яті.	
tktvno	Номер версії формату білета.
Realm	Ім'я області (домена), де генерований білет.
Sname	Ім'я сервера.
Інші поля шифруються за допомогою секретного ключа сервера.	
Flags	Прапори білета.
Key	Сеансовий ключ.
Crealm	Ім'я області (домена) клієнта.
Cname	Ім'я клієнта.
Transited	Список областей Kerberos, що приймали участь в автентифікації клієнта, якому виданий даний білет.
Authtime	Час первісної автентифікації клієнта.
Starttime	Час набрання білетом чинності.
Endtime	Час витікання терміну дії білета.
renewtill	Найбільше значення поля *endtime*, що може бути задане за допомогою прапора RENEWABLE (поле необов'язкове).
Caddr	Одна або кілька адрес, з яких може використовуватися даний білет.
Authorization-data	Атрибути привілеїв клієнта.

Таблиця 2.2 – Прапори білета

Прапор	Опис
FORWARDABLE	Указує, що на підставі даного білета TGT служба видачі білетів може генерувати новий білет TGT з іншою мережевою адресою (поле є тільки у білетах TGT).
FORWARDED	Указує на те, що даний білет TGT був переадресований або генерований на основі іншого білета TGT, що пройшов переадресацію.
PROXIABLE	Указує, що служба видачі білетів може генерувати білети, мережева адреса яких буде відрізнятися від наведеного в даному квитку TGT (поле є тільки у білетах TGT).
PROXY	Указує на те, що мережева адреса в даному квитку відрізняється від адреси, наведеної у квитку TGT, на підставі якого він виданий.
RENEWABLE	Використовується в сполученні з полями *endtime* і *renew-till*, дозволяючи періодичне відновлення службою KDC білетів з підвищеним терміном дії.
INITIAL	Указує, що даний білет є білетом видачі білетів (поле є тільки у білетах TGT).

Є п'ять учасників Протоколу Kerberos (рисунок 2.1):

U: Користувач U виступає, ініціалізує за допомогою пароля відповідний процес C(U). Користувач повинен знати свій одноразовий пароль на звертання до протоколу Kerberos.

C: Клієнт – це процес, що використовує ресурс мережі з доручення користувача U. Для його запуску необхідне свідчення про допуск користувача до протоколу Kerberos, що й повідомляється йому паролем користувача.

AS: Сервер автентифікації (Authentication Server) – процес, що видає по запиту AS_REQ (Kerberos Authentication Service Request – запит до служби

автентифікації Kerberos від С) мандат TGT на право одержання білетів TKT.

TGS: Сервер, що видає по запиту TGS_REQ (Kerberos Ticket-Granting Service Request – запит до служби видачі білетів Kerberos) від С білет TKT (Ticket Kerberos Ticket) на доступ до ресурсу.

S: Сервер додатків – процес, що надає клієнтові З по його запиту AP_REQ (Kerberos Application Request – запит додатка Kerberos) дозвіл на використання мережевого ресурсу (AP_REP (Kerberos Application Reply – відповідь додатка Kerberos)).

AS і TGS утворять центр розподілу ключів (KDC (Key Distribution Center). Поділ на два сервери дозволяє розвантажити AS від роботи з видачі сеансових ключів.

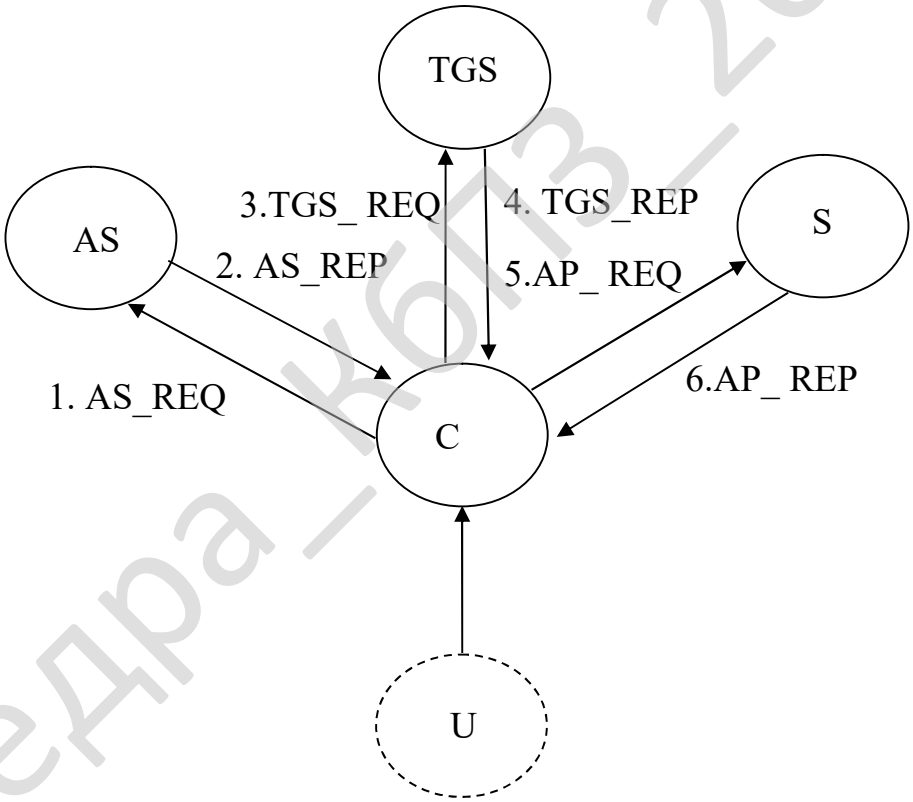

Рисунок 2.1 – П'ять учасників і три протоколи обміну в протоколі Kerberos

Підпротоколи протоколу Kerberos:

Протокол автентифікації AS Exchange – Authentication Service Exchange, взаємодіючий між клієнтом С и сервером автентифікації AS.

Використовується KDC для передачі клієнтові сеансового ключа реєстрації й білета TGT (Ticket Granting Ticket) на одержання білетів-автентифікаторов.

Протокол TGS Exchange – Ticket-Granting Service Exchange, взаємодія між клієнтом С і сервером TGS. Служить для розсилання службових сеансових ключів і сеансових ключів самої служби KDC. Протокол містить схему й відповідну програму розподілу ключів, що дозволяє обчислити ключі для кожної пари клієнт-сервер додатків.

Протокол Client/Server Authentication Application Exchange (AP Exchange) автентифікованого обміну клієнта С із сервером S додатків.

Всі протоколи працюють у дві сторони.

Для AS Exchange:

С(AS: AS_REQ (запит білета TGT);

AS (C: AS_REP (білет TGT).

Для TGS Exchange:

С(TGS: TGS_REQ (запит білета-автентифікатора TKT);

TGS (C: TGS_REP (TKT).

Для Application Exchange:

С(AP: AP_REQ (запит мережевого ресурсу);

AP (C: AP_REP (дозвіл на використання мережевого ресурсу).

Попередній розподіл ключів – одна з найважливіших проблем інформаційного обміну. У великих мережах виникає потреба у великій кількості ключів для зв'язку між абонентами мережі. Попередній розподіл ключів дозволяє істотно зменшити об'єм ключової інформації, переданої по секретних каналах, зберігаючи при цьому можливість з'єднання абонентів мережі, що входять у певні коаліції.

Суть попереднього розподілу ключів полягає в тому, що абонентам мережі розподіляються не самі ключі, а деякий допоміжний матеріал, що займає (у сукупності) менший об'єм.

На підставі даного матеріалу кожний абонент мережі по деякому

алгоритмі може самостійно обчислити ключ, необхідний для зв'язку з деякою групою абонентів (*привілейованою* групою); те ж саме можуть зробити всі члени цієї групи. У той же час, певні групи абонентів (*заборонені* групи), об'єднавши допоміжний матеріал, не можуть одержати яку-небудь інформацію про цей ключ, якщо цей абонент не входить у жодну із груп, що об'єдналися.

Відомо кілька схем попереднього розподілу ключів, наприклад, схема Блома (Blom Scheme), схема Блундо (Blundo *et al* Scheme), схема Фиат-Наора (Fiat-Naor Scheme).

Розглянемо методи побудови схем KDP (Key Distribution Pattern) – *схеми розподілу системних ключів*. У даній схемі кожному абонентові мережі розподіляються системні ключі, обирані з деякої множини.

Визначення 1: KDP(***P***,***F***)-схемою, де ***P*** і ***F*** – це сімейства підмножин множини $U = \{1, \ldots, n\}$, називається всяке сімейство $\boldsymbol{B}=\{B_1, \ldots, B_k\}$ підмножин множини U, що задовольняє умові:

$$\forall P \in \boldsymbol{P}\ \forall F \in \boldsymbol{F}, F \mid P = \varnothing: \quad \{B_s : P \subseteq B_s,\ F \mid B_s = \varnothing\} \neq \varnothing. \qquad (2.1)$$

Визначення 2: KDP(***P***,***F***)-схемою, де ***P*** і ***F*** – це сімейства підмножин множини $U = \{1, \ldots, n\}$, називається всяке сімейство $\boldsymbol{S}=\{S_1, \ldots, S_n\}$ підмножин кінцевої множини $\Psi \subseteq \Omega$ ($|\Psi|=k$), що задовольняє умові:

$$\forall P \in \boldsymbol{P}, F \in \boldsymbol{F}, P \cap F = \varnothing: \quad \bigcap_{i \in P} S_i \not\subset \bigcup_{j \in F} S_j . \qquad (2.2)$$

Тут U – множина абонентів мережі, ***P*** і F-***F***– привілейоване й заборонене сімейства відповідно, Ω – це множина можливих системних ключів, а Ψ – множина системних ключів мережі, що розподіляються абонентам.

У даній роботі доведена еквівалентність двох визначень.

Ключ k, необхідний для зв'язку з деякою групою абонентів, обчислюється абонентом мережі як значення деякої геш-функції на множині системних ключів, загальних для всіх абонентів даної групи:

$$k_P = h\left(\bigcap_{i \in P} S_i\right), \text{ для будь-якого } P \in \boldsymbol{P}. \qquad (2.3)$$

Якщо множина коаліцій привілейованих користувачів включає всі

підмножини множини U з g користувачів, а множина заборонених коаліцій – всі підмножини з w користувачів, то схема KDP($\boldsymbol{P}$,$\boldsymbol{F}$) позначається KDP(g,w).

Розглянемо два відомих тривіальних методи побудови KDP-схем.

Перший метод полягає в тому, що всім абонентам привілейованої коаліції P довірений центр ТА (Trusted Authority) виділяє один системний ключ, що не знають інші абоненти:

$$\forall P \in \boldsymbol{P}\ \forall i \in P : \omega^P \in S_i, \forall j \notin P : \omega^P \notin S_j . \qquad (2.4)$$

При цьому кількість системних ключів |Ψ| у схемі, побудованої даним методом, буде k=|$\boldsymbol{P}$|. Помітимо, що даний метод не залежить від сімейства $\boldsymbol{F}$ і, отже, можна вважати, що $\boldsymbol{F}$=2^U. Далі KDP-схеми, побудовані даним методом будемо позначати KDP($\boldsymbol{P}$,$\cdot$).

Другий метод полягає в тому, що ТА виділяє один системний ключ кожному абонентові, що не входить у заборонену коаліцію F:

$$\forall F \in \boldsymbol{F}\ \forall i \notin F : \omega^F \in S_i, \forall j \in F : \omega^F \notin S_j . \qquad (2.5)$$

При цьому ТА повинен виділити один системний ключ всім абонентам мережі:

$$\omega^0 \in S_i\ \forall i \in U . \qquad (2.6)$$

Даний системний ключ необхідний для забезпечення зв'язку між коаліціями абонентів, які перетинаються з усіма забороненими коаліціями. Наприклад, для P=U. Кількість системних ключів |Ψ| у схемі, побудованої даним методом, буде k=|$\boldsymbol{F}$|+1.

Даний метод не залежить від сімейства $\boldsymbol{P}$ і, отже, можна вважати, що $\boldsymbol{P}$=2^U.KDP-схеми, побудовані даним методом позначаються KDP($\cdot$, $\boldsymbol{F}$) .

Дані тривіальні методи раціонально використовувати при невеликих потужностях сімейств $\boldsymbol{P}$ і $\boldsymbol{F}$.

Приклад KDP(2,1)-схеми для 4 абонентів мережі, що використовує 4 ключі, наведений у таблиці 2.3.

У таблиці рядкам таблиці відповідають абоненти мережі, стовпцям –

системні ключі. На перетинанні *i-го* рядка й *j-го* стовпця стоїть 1, якщо *j-ий* системний ключ належить *i-му* абонентові.

Таблиця 2.3 – KDP(2,1)-схема для n=4 абонентів.

	ψ_1	ψ_2	ψ_3	ψ_4
A_1	1	1	1	
A_2	1	1		1
A_3	1		1	1
A_4		1	1	1

Існують відомі методи, що дозволяють побудувати окремі випадки схем розподілу системних ключів, наприклад, імовірнісний і засновані на ортогональних масивах, перпендикулярних масивах, урівноважених неповних блок-схемах. У цих випадках будь-які g абонентів здатні обчислити загальний ключ, і ніякі інші w абонентів не здатні довідатися яку-небудь інформацію про цей ключ.

У літературі [10-20] обґрунтовано висновок переваги імовірнісних методів побудови схем попереднього розподілу ключів над детермінованими.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об’єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об’єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об’єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами

даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис методу побудови схем попереднього розподілу ключів

У цьому розділі наведені приклади поліпшення схем попереднього розподілу системних ключів за рахунок аналізу привілейованого й забороненого сімейств.

Визначення: Нехай:

$\sum^1$ – множина всіх можливих KDP($\boldsymbol{P}_1$,$\boldsymbol{F}_1$)-схем.

$\sum^2$ – множина всіх можливих KDP($\boldsymbol{P}_2$,$\boldsymbol{F}_2$)-схем.

$\boldsymbol{S}^1$ – KDP($\boldsymbol{P}_1$,$\boldsymbol{F}_1$)-схема ($\boldsymbol{S}^1 \in \sum^{1)}$.

$\boldsymbol{S}^2$ – KDP($\boldsymbol{P}_2$,$\boldsymbol{F}_2$)-схема ($\boldsymbol{S}^2 \in \sum^{2)}$.

Схеми $\boldsymbol{S}^1$ і $\boldsymbol{S}^2$ є взаємозамінними, якщо $\boldsymbol{S}^1 \in \sum^2$ і $\boldsymbol{S}^2 \in \sum^{1.}$

Для взаємозамінних схем по визначенню KDP-схеми виконано:

$$\forall P \in \boldsymbol{P}_1, F \in \boldsymbol{F}_1, P \cap F = \varnothing: \ \bigcap_{i \in P} S_i^2 \not\subset \bigcup_{j \in F} S_j^2, \tag{3.1}$$

$$\forall P \in \boldsymbol{P}_2, F \in \boldsymbol{F}_2, P \cap F = \varnothing: \ \bigcap_{i \in P} S_i^1 \not\subset \bigcup_{j \in F} S_j^1 . \tag{3.2}$$

Помітимо, що відношення взаємозамінності є симетричні і рефлексивні, але не є транзитивні, тобто з того, що схеми $\boldsymbol{S}^1$ і $\boldsymbol{S}^2$ взаємозамінні, $\boldsymbol{S}^2$ і $\boldsymbol{S}^3$ взаємозамінні, не слідує, що $\boldsymbol{S}^1$ і $\boldsymbol{S}^3$ взаємозамінні. Відношення взаємозамінності є відношенням толерантності.

Твердження 1. Якщо $\boldsymbol{F}_1 \subseteq \boldsymbol{F}$, де $\boldsymbol{F}_1$ це сімейство всіх підмножин множини U потужності w, причому $\forall P \in \boldsymbol{P}: |P| + w < n$ $(\max\limits_{P \in \boldsymbol{P}} |P| + w < n)$, то KDP($\boldsymbol{P}$,$\boldsymbol{F}$) і KDP($\boldsymbol{P}, \boldsymbol{F}'$)-схеми є взаємозамінними, де $\boldsymbol{F}'$ – це об'єднання сімейства $\boldsymbol{F}$ і всіх підмножин множини U потужність яких не перевершує w.

Твердження 2. Якщо існує сімейство $\boldsymbol{P}_1 \subseteq \boldsymbol{P}$, таке що $\boldsymbol{P}_1$ це сімейство всіх підмножин множини U потужності g, причому $\forall F \in \boldsymbol{F} : |F| + g < n$, то KDP($\boldsymbol{P}$,$\boldsymbol{F}$) і KDP($\boldsymbol{P}'$,$\boldsymbol{F}$)-схеми є взаємозамінними, де $\boldsymbol{P}'$ – це об'єднання сімейства $\boldsymbol{P}$ і всіх підмножин множини U, потужність яких не перевершує g.

Твердження 3. Якщо $g+w<n$, то схеми KDP(g,w) і KDP($\leq g$, $\leq w$) є взаємозамінними.

Для імовірнісного методу запропонована більше низька оцінка кількості системних ключів, необхідних для побудови схеми.

На основі вивчених методів запропонований метод побудови загального випадку схем розподілу системних ключів.

Суть імовірнісних методів полягає в тому, що формується деяким випадковим образом таблиця. Далі перевіряється, чи є сформована таблиця KDP-схемою. Якщо немає – то таблиця формується заново.

Розглянемо імовірнісний метод побудови KDP($\boldsymbol{P}$,$\boldsymbol{F}$)-схем.

Нехай елементи множини Ψ пронумеровані: $\{\psi_1, \psi_2, \ldots, \psi_k\}$.

Позначимо $X_{is} = \begin{cases} 1, & \text{якщо } \psi_s \in S_i \\ 0, & \text{якщо } \psi_s \notin S_i \end{cases}$, s=1, …, k; i=1, …, n...

Таблиця X заповнюється в такий спосіб:

$$X_{is} = \begin{cases} 1, & \text{з ймовірністю } p \\ 0, & \text{з ймовірністю } 1-p \end{cases}. \qquad (3.3)$$

Верхня оцінка ймовірності того, що KDP-схема не буде побудована імовірнісним методом:

$$E = E(k,p) = \sum_{P \in \boldsymbol{P}} \sum_{\substack{F \in \boldsymbol{F} \\ P \cap F = \varnothing}} \prod_{s=1}^{k} \left(1 - \prod_{i \in P} p \prod_{j \in F} (1-p) \right) = \sum_{P \in \boldsymbol{P}} \sum_{\substack{F \in \boldsymbol{F} \\ P \cap F = \varnothing}} \left(1 - p^{|P|} (1-p)^{|F|} \right)^k \qquad (3.4)$$

Розглянуто імовірнісний метод, що дозволяє побудувати KDP$(\boldsymbol{P}^g, \boldsymbol{F}^w)$ - схеми, де:

$$\boldsymbol{P}^g = \{P : P \in \boldsymbol{P}, |P| = g\}, \ \boldsymbol{F}^w = \{F : F \in \boldsymbol{F}, |F| = w\}. \qquad (3.5)$$

Мінімізуючи $E(k, p)$ по p, маємо:

$$p_0 = \frac{g}{w+g}. \tag{3.6}$$

Кількість ключів $k_0^{g,w}$, необхідних для побудови такої схеми обчислюється по формулі:

$$k_0^{g,w} = \left\lfloor \frac{(g+w)^{g+w}}{g^g w^w} \ln\left(\frac{\sum\limits_{P \in \boldsymbol{P}^g} \sum\limits_{\substack{F \in \boldsymbol{F}^w \\ P \cap F = \varnothing}} 1}{1 - \mathrm{E}} \right) \right\rfloor + 1. \tag{3.7}$$

Уведено поняття об'єднання KDP-схем і запропоновані способи зменшення об'єму ключового матеріалу за рахунок комбінування різних методів побудови схем KDP (або схем, одержуваних одним методом, але з різними параметрами).

Визначення. Нехай:

$\{S_1^1, \ldots, S_n^1\}$ – KDP($\boldsymbol{P}_1$,$\boldsymbol{F}_1$)-схема.

$\{S_1^2, \ldots, S_n^2\}$ – KDP($\boldsymbol{P}_2$,$\boldsymbol{F}_2$)-схема.

При цьому $\Psi_1 \bigcap \Psi_2 = \varnothing$.

Тоді об'єднанням KDP-схем KDP($\boldsymbol{P}_1$,$\boldsymbol{F}_1$)$\bigcup$ KDP($\boldsymbol{P}_2$,$\boldsymbol{F}_2$) є сімейство $\{S_1, \ldots, S_n\}$: $S_i = S_i^1 \bigcup S_i^2$.

Позначимо $G = \{g : \boldsymbol{P}^g \neq \varnothing\}$, $W = \{w : \boldsymbol{F}^w \neq \varnothing\}$.

Тоді:

$$\boldsymbol{P} = \bigcup_{g \in G} \boldsymbol{P}^g, \ \boldsymbol{F} = \bigcup_{w \in W} \boldsymbol{F}^w \ \text{і} \ \mathrm{KDP}(\boldsymbol{P}, \boldsymbol{F}) = \bigcup_{w \in W, g \in G} \mathrm{KDP}(\boldsymbol{P}^g, \boldsymbol{F}^w). \tag{3.8}$$

При цьому можна використовувати різні методи побудови KDP$(\boldsymbol{P}^g, \boldsymbol{F}^w)$ - схем.

Розглянемо комбінування імовірнісних і тривіальних методів побудови схем попереднього розподілу системних ключів.

Якщо виконуються умови:

$$H_1(g,w)=\{\sum_{g\in G}\min\{k_0^{g,w},\left|\boldsymbol{P}^g\right|\}\geq\left|\boldsymbol{F}^w\right|\ \},\qquad(3.9)$$

або

$$H_2(g,w)=\{\sum_{w\in W}\min\{k_0^{g,w},\left|\boldsymbol{F}^w\right|\}\geq\left|\boldsymbol{P}^g\right|\ \},\qquad(3.10)$$

то для побудови KDP$(\boldsymbol{P}^g,\boldsymbol{F}^w)$ -схеми раціонально використовувати тривіальні методи, інакше – імовірнісні.

Таким чином, KDP(***P***,***F***)-схема:

$$\bigcup_{\substack{w\in W,g\in G,\\ \neg H_1(g,w),\\ \neg H_2(g,w)}}\mathrm{KDP}(\boldsymbol{P}^g,\boldsymbol{F}^w)\cup\bigcup_{\substack{w\in W,\\ H_1(w,g)}}\mathrm{KDP}(\cdot,\boldsymbol{F}^w)\cup\bigcup_{\substack{g\in G,\\ H_2(g,w)}}\mathrm{KDP}(\boldsymbol{P}^g,\cdot)\qquad(3.11)$$

Кількість системних ключів |Ψ|, необхідних для побудови схеми:

$$k_0=\sum_{\substack{w\in W,g\in G,\\ \neg H_1(g,w),\\ \neg H_2(g,w)}}k_0^{g,w}+\sum_{\substack{w\in W,\\ H_1(w,g)}}\left|\boldsymbol{F}^w\right|+\sum_{\substack{g\in G,\\ H_2(g,w)}}\left|\boldsymbol{P}^g\right|\qquad(3.12)$$

Імовірнісний метод побудови схеми попереднього розподілу ключів з геш-функцією.

Визначення. HAKDP(***P***,*F*,*L*)-схемою, де ***P*** і ***F*** – це сімейства підмножин множини $U = \{1, \ldots, n\}$, називається всяка пара сімейств (***S***,***D***), ***S***=$\{S_1,\ldots,S_n\}$ підмножин кінцевої множини Ψ⊆Ω (|Ψ|=*k*) і ***D***=$\{D_1,\ldots,D_n\}$ підмножин множини $\{1,\ldots,L\}$, причому $|D_t| = |S_t|$ для t=1,…,n, задовольняючій умові: $\forall P\in\boldsymbol{P},F\in\boldsymbol{F},P\cap F=\varnothing:\bigcap_{i\in P}S_i\not\subset\bigcup_{j\in F}S_j$ або не виконана умова $D_{F,P}\leq D_P$, де D_P – набір значень $\max_t(D_i(t))$ відповідним співпадаюніж елементам множин S_i з P і $D_{F,P}$ є набір значень $\min_t(D_i(t))$ всіх тих елементів з F, які відповідають співпадаюніж елементам множин S_i з F.

Зауваження. HAKDP(***P***,***F***,0)-схема є KDP(***P***,***F***)-схемою.

Приклад HAKDP(2,1,1)-схеми для n=4 абонентів і k=3 ключів:

S_1={1,2,3}, D_1=(1,1,1),

S_2={1,2}, D_2=(0,0),

S_3={1,3}, D_3=(0,0),

S_4={2,3}, D_4=(0,0)

Нехай, як і у випадку KDP-схем, таблиця X отримана деяким випадковим способом $\Pr\{X_{is}=1\}=p_0$. Таблиця $\boldsymbol{D}$ також отримана випадковим способом $\Pr\{D_t(s)=a\}=\frac{1}{L}$, $a \in \{1,...,L\}$. При цьому p_0 обчислюється по формулі:

$$p_0 = \frac{2}{3(1+P_L)}, \tag{3.13}$$

кількість ключів, необхідних для побудови HAKDP(2,1,L)-схеми:

$$k = -\frac{\log(n(n-1)(n-2))}{\log(1-p^2(1-p)-p^3P_L)}, \tag{3.14}$$

де P_L – імовірність того, що $D_{rs} < D_i(s)$ і $D_r(s) < D_j(s)$.

Дану ймовірність можна порахувати по формулі:

$$P_L = \sum_{m=0}^{L} \frac{1}{L+1}\left(\frac{L-i}{L+1}\right)^2 = \frac{2L^2+L}{6L^2+12L+6}. \tag{3.15}$$

Переваги використання KDP і HAKDP-схем випливають із даних, представлених у таблиці 3.1 на прикладі KDP(10^4, 173) і HAKDP(10^4, 114, 10)-схем.

Таблиця 3.1 – Порівняння централізованої KDP і HAKDP-схем

	Без попереднього розподілу	KDP(10^4,173)-схема	HAKDP(10^4,114,10)-схема
Середня довжина пакета	9 999	115	71
Число переданих ключів	99 990 000	1 150 000	710 000

Пропонується три способи організації безпечної мережі на основі протоколу Kerberos і схем попереднього розподілу ключів.

Перший спосіб припускає використання стандартного протоколу Kerberos,

у якому як сервер додатків виступає абонент мережі, якому адресується повідомлення.

Другий спосіб припускає використання стандартного протоколу Kerberos для попереднього розподілу ключового матеріалу в мережі, відповідно до KDP-схеми. При цьому KDP-схема й ключовий матеріал формується на стороні сервера додатків.

Третій спосіб припускає використання модернізованого протоколу Kerberos для розподілу ключового матеріалу. При цьому KDP-схема й ключовий матеріал формуються на стороні сервера TGS.

Розглянемо нецентралізований розподіл ключової інформації. У складних мережах, що складаються з декількох доменів, використання централізованого розподілу ключової інформації може бути важко.

Опишемо технологію захищених комунікацій у комп'ютерній мережі, з використанням ключової інформації, попередньо розподіленої в її сегментах. Хоча б один з елементів у кожному сегменті відіграє роль сервера додатків. Інші елементи сегмента представляють мережевих клієнтів. Мережеві сервери додатка є довіреними вузлами для всіх елементів комп'ютерної мережі. Вони зашифровують і розшифровують конфіденційну інформацію користувачів, коли відповідний шифротекст одного сегмента передається користувачеві з іншого сегмента. При цьому використовуються ключі, що обчислюються по пачках, розподіленим елементам одного сегмента, для розшифрування й іншого для зашифрування.

Захищені комунікації в мережі здійснюються з використанням ключів, що обчислюються на підставі часток ключової інформації, виділюваних окремим елементам або різним парам елементів того самого домена (рисунок 3.1).

Дана технологія дозволяє зменшити час обчислення схем попереднього розподілу ключів, а також об'єм ключової інформації, переданої по секретних каналах від генератора до елементів комп'ютерної мережі.

Передбачаються наступні особливості мережі:

1) комп'ютерна мережа складається з N сегментів (або доменів) $D_1,\ldots, D_N$;

2) хоча б один учасник кожного домена виконує роль мережевого сервера додатків NAS (Network Application Server), інші елементи є клієнтами мережі.

3) кожний NAS є елементом двох сусідніх сегментів, він одержує свої частки ключової інформації, розподіленої в обох сегментах;

4) граф, вершини якого відповідають сегментам, а ребра – мережевим серверам додатків NAS зв'язний;

5) кожний учасник може передавати зашифровану інформацію будь-якому іншому учасникові по відкритих каналах;

6) мережеві сервери додатків NAS є довіреними вузлами для всіх елементів комп'ютерної мережі;

7) кожний мережевий сегмент (або домен) D_j, j=1,…,N, прикріплений до певного довіреного центра TA_j;

8) кожний довірений центр TA_j, j=1,…,N, виробляє схему попереднього розподілу ключів для прикріпленого сегмента й розподіляє між його елементами пакети обчисленої ключової інформації;

9) генерація схем розподілу ключів і доставка ключової інформації для елементів різних сегментів здійснюється незалежно й, можливо, одночасно.

10) конфіденційна комунікація між учасниками комп'ютерної мережі організується з використанням зашифрування-розшифрування й забезпечення цілісності тих самих або різних пакетів ключової інформації, що належать одному або декільком учасникам того самого сегмента.

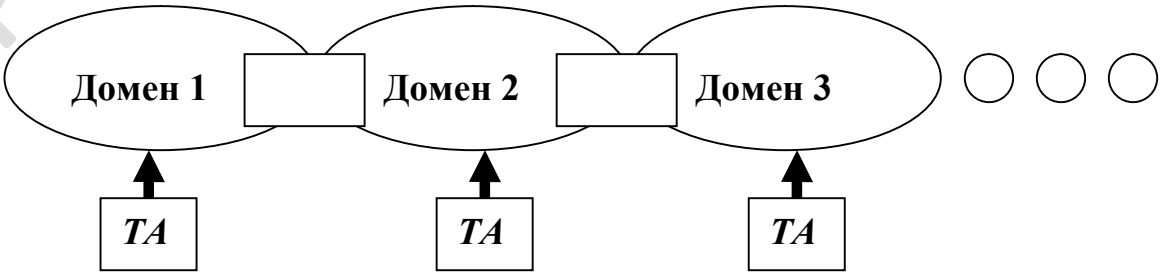

Рисунок 3.1 – Нецентралізований попередній розподіл ключів

Розглянемо окремий випадок мережі, у якій кожний абонент мережі може передати інформацію іншому абонентові мережі.

Розподіл ключової інформації здійснюється незалежно у всіх доменах мережі. Для цього в кожному домені будується KDP(2,1)-схема. Такі схеми можуть мати однакову або різні структури, але в кожній з них елементи ключової інформації, що розподіляється, виробляються випадково й незалежно.

Нехай по завершенню етапу попереднього розподілу ключів (централізованому для кожного домена образом, але незалежним у кожному домені) всі учасники комп'ютерної мережі мають пакети ключової інформації. Це дозволяє їм обчислити ключі для конфіденційного обміну в межах одного домена.

Для конфіденційної передачі інформації від абонента А абонентові В того самого домена, абонент А шифрує інформацію ключем k_{AB}, що будується на основі ключової інформації абонента А і KDP-схеми даного домена.

Для конфіденційної передачі інформації від абонента А абонентові В сусіднього (що має з доменом абонента А загальний сервер додатка С) домена, А передає С інформацію, зашифровану ключем k_{AC}. Сервер додатка С зашифровує інформацію за допомогою ключа k_{CB}, потім передає інформацію абонентові В, розшифровану ключем k_{AC}.

Для конфіденційної передачі інформації від абонента А абонентові В домена, що не має з доменом елемента А загального сервера додатків, будується шлях від абонента А до абонента В через сервера додатків C_1, C_2, …, C_l... Далі абонент А передає серверу додатків C_1 інформацію, зашифровану ключем k_{AC_1}. Сервер додатків C_1 зашифровує інформацію за допомогою ключа $k_{C_1C_2}$, потім передає інформацію серверу додатків C_2, розшифровану ключем k_{AC_1}, і так далі. Наприкінці сервер додатків C_l передає абонентові В інформацію, зашифровану ключем k_{C_lB}.

Помітимо, що відкритий текст відправника доступний для атак зловмисника через сервери додатків NAS, вони є довіреними вузлами для всіх

учасників інформаційного конфіденційного обміну. Якщо шифратор відповідає властивості перестановочності (як, наприклад, блоковий шифратор у режимі гаммування), то розшифруванню в довірених вузлах зв'язку може передувати вторинне зашифрування, що було вище.

Для скорочення операцій розшифрування й зашифрування у вузлах зв'язку можливий наступний варіант: абонент А передає абонентові В ключ K_{AB}, обчислений на основі його ключового пакета, а потім здійснює передачу інформації, зашифрованої ключем K_{AB}.

Ключ зашифрування є значенням геш-функції, застосованої до ідентифікатора абонента В. Передача ключа K_{AB} і інформації здійснюється описаним вище способом.

У результаті виконання магістерської роботи досліджений також спосіб організації конференц-зв'язку коаліції абонентів з різних доменів в умовах нецентралізованого розподілу ключової інформації.

Модифікація протоколу Kerberos для генерації й розподілу ключової інформації

Генерація й розподіл ключової інформації для безпечної мережі може бути організована з використанням модифікації протоколу Kerberos. Запропонована модифікація протоколу Kerberos для генерації й розподілу ключової інформації в комп'ютерній мережі відрізняється тим, що в серверах TGS обчислюються пакети ключової інформації, що доставляються клієнтам із сервера додатків у складі посилок TGT.

Нехай є один або більше серверів автентифікації (AS) і, принаймні, така ж кількість серверів видачі квитків (TGS).

Кожний сервер TGS відповідає одному серверу AS. Кожний домен мережі зв'язується із сервером TGS і абоненти домена прив’язуються до цього TGS. Якщо сервер додатків NAS утримується в декількох доменів, то він прив’язується до всіх TGS приналежних даним доменам. Всі абоненти мережі підтримують службу одноразових паролів OTPS (One-Time Password Service) з декількома AS,

до яких прив'язані відповідні TGS.

Сервер додатків NAS підтримує службу одноразових паролів однієї або більше OTPS. Абонент A_i і сервер автентифікації AS_j з OTPS мають одноразовий пароль k_{A_i,AS_j}.

На етапі ініціалізації (рисунок 3.2) кожний сервер AS_j обчислює ключі k_{A_i,TGS_t} для зв'язку з TGS_t і прив'язаних до нього абонентів A_i, k_{NAS_p,TGS_t} для зв'язку TGS_t і прив'язаних до нього NAS_p.

Припустимо, що в кожного AS_j є ключ k_{AS_j,TGS_t} для зв'язку із прив'язаними до нього TGS_t і кожний TGS будує KDP(***P***,***F***)-схему для відповідного домена мережі: для кожного абонента A_i або NAS_p цього домена обчислюються пакети S_i, S_s.

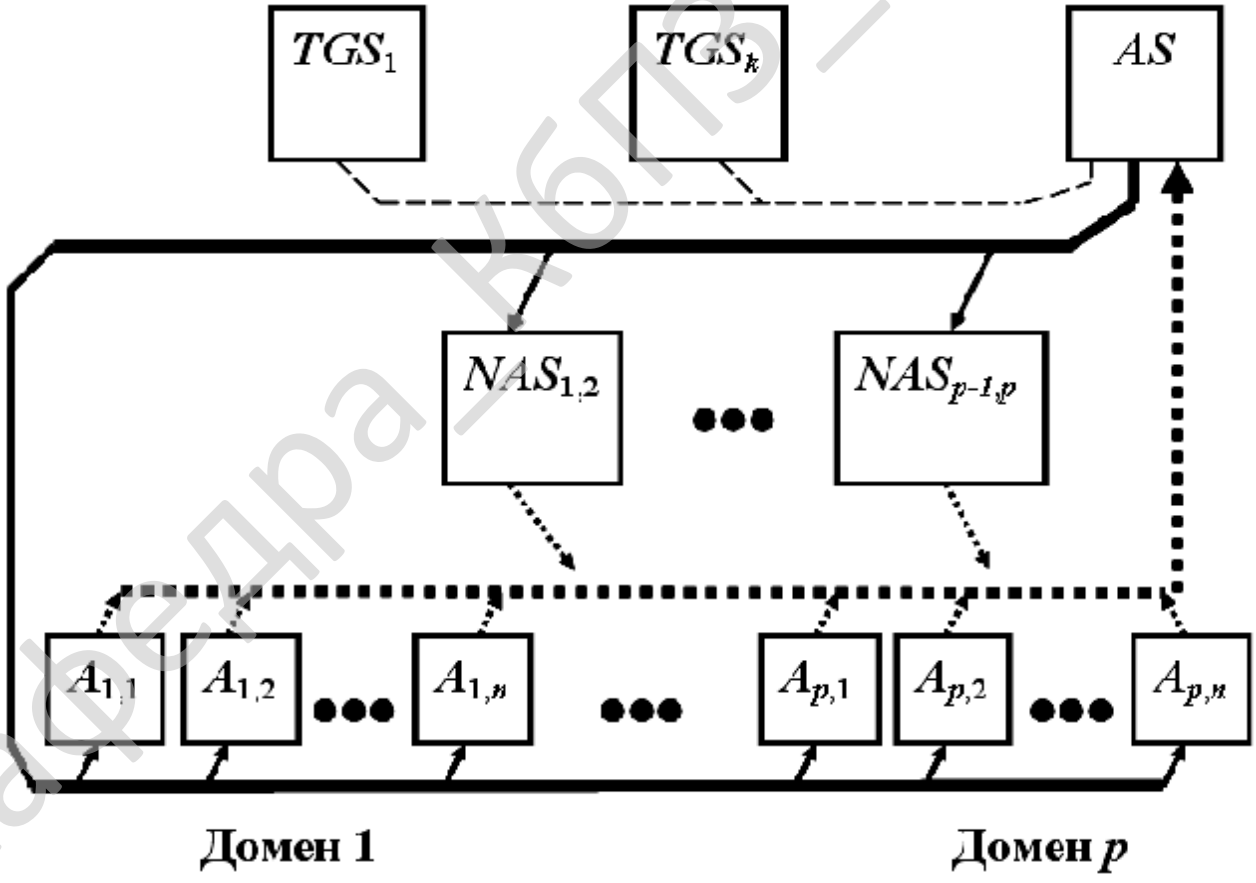

Рисунок 3.2 – Протокол обміну із сервером автентифікації з метою одержання дозволу на видачу ключової інформації

Тепер кожний учасник A_i підтримуючу службу одноразового пароля з відповідним сервером автентифікації AS_j, може одержати свій пакет K_i ключової інформації, ініціалізуя й виконуючи протоколи:

a) протокол обміну із сервером автентифікації, з метою одержання дозволу TGT на одержання ключової інформації (рисунок 3.2);

b) протокол обміну із сервером TGS, з метою одержання ключової інформації (рисунок 3.3);

Для безпечного обміну інформації між абонентами мережі використовується:

c) протокол комунікації абонентів мережі, у тому числі через сервер додатків NAS (рисунок 3.4).

Дано оцінки ключової інформації, необхідної для організації захищених комунікацій, для мереж різних типів.

Дані, представлені в таблиці 3.2 показують переваги використання нецентралізованих 10(KDP(1002, 56) і 10(HAKDP(1002, 56, 10)-схем.

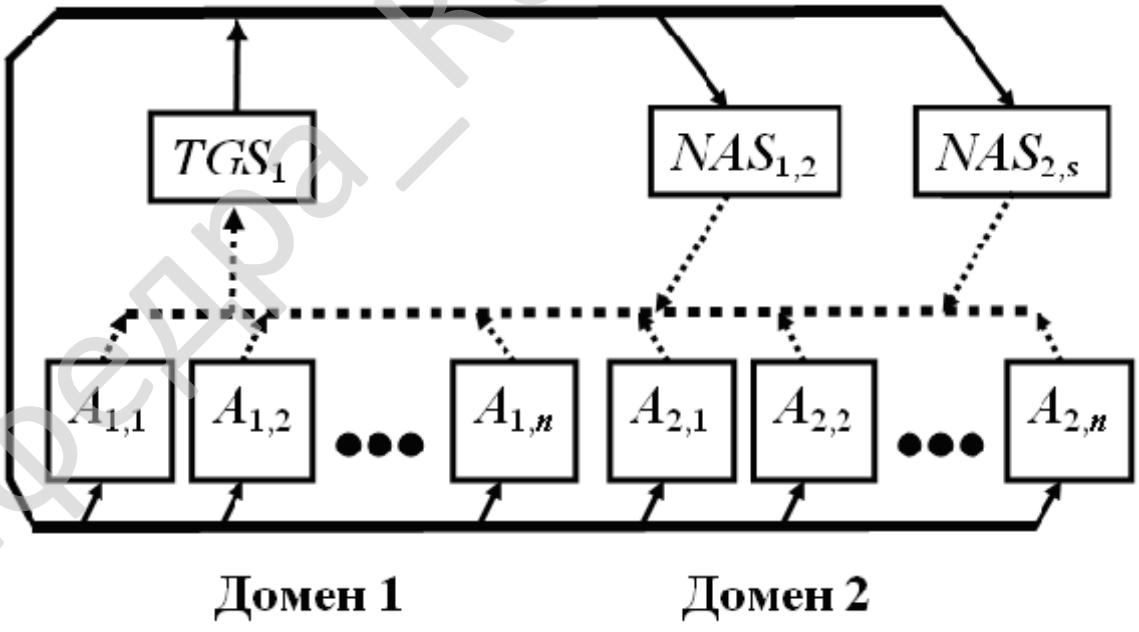

Рисунок 3.3 – Протокол обміну із сервером *TGS* з метою одержання ключового матеріалу абонентами мережі *A* і серверами додатків *NAS*

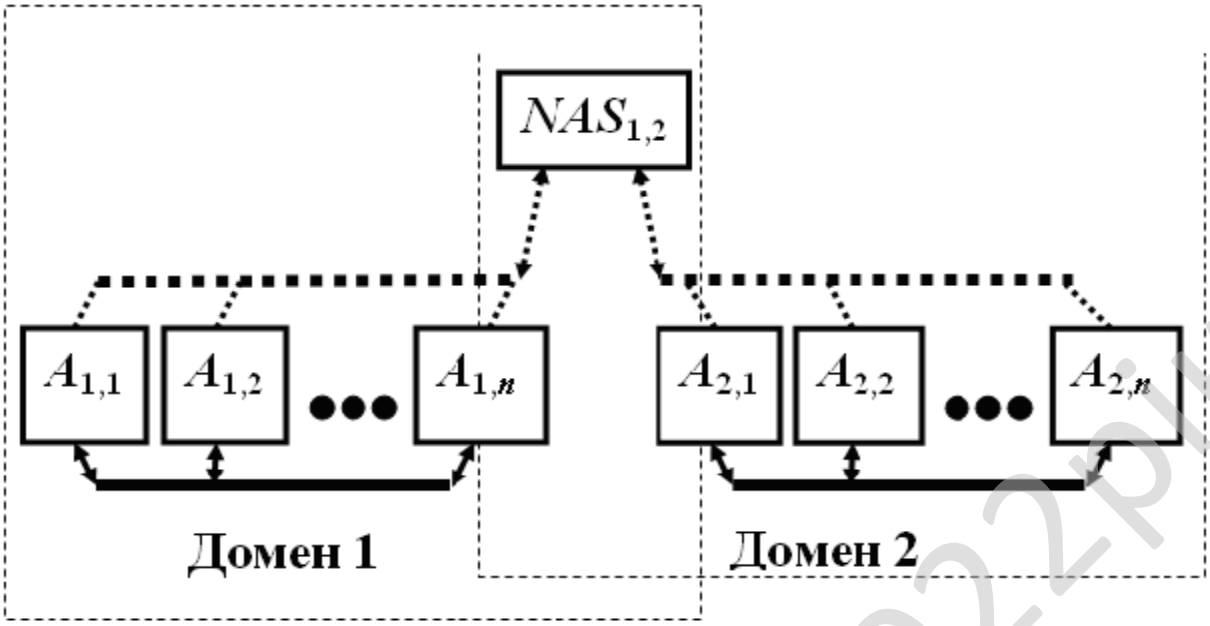

Рисунок 3.4 – Протокол комунікацій абонентів мережі, у тому числі через сервер додатків *NAS*

Як баніжо, у нецентралізованій мережі в порівнянні із централізованої (таблиця 3.1), число ключів, що пересилаються від ТА учасникам, скорочується. Застосування схем з гешуванням приводить до ще більшого скорочення середньої довжини пакетів ключової інформації й об'єму ключової інформації, переданої від довіреного центра ТА.

Таблиця 3.2 – Порівняння нецентралізованої KDP і HAKDP

	Без попереднього розподілу	KDP(10^4,173)-схема	HAKDP(10^4,114,10)-схема
Середня довжина пакета	1 001	37	28
Число переданих ключів	10 030 020	370 740	280 560

Також досліджуються особливості побудови програмних засобів попереднього розподілу ключів. Показано, що при цьому перевага віддається

імовірнісним методам двухетапного синтезу схем.

На першому етапі випадково генерується таблиця розподілу ключів, а на другому – перевіряється її відповідність умовам необхідної схеми. Аналітично обґрунтовані параметри керування імовірнісного етапу. Працездатність запропонованих методів підтверджена експериментами з використанням розроблених програмних засобів при різних параметрах вимог до мережі.

3.2 Розробка структурної схеми

Структурна схема розробленої системи показана на рисунку 3.5. Для реалізації мережі WAN було вибрано технологію приватної мережі на орендованих каналах.

Мережа складається з головного офісу та декількох філіалів. У приміщенні головного офісу знаходяться наступні сервери:

1. Сервер автентифікації.
2. Сервер квитанцій (білетів).
3. Ресурсний сервер.

У приміщеннях філіалів знаходяться:

1. Ресурсні сервери.
2. Робочі станції (хости).

Орендовані територіальні канали прокладаються провайдером транспортних територіальних послуг у його первинній мережі FDM, PDH, SDH або мережі з інтегральними послугами ISDN. При оренді каналу в таких мережах підприємство ділить пропускну здатність магістральних каналів і комутаторів цієї мережі з іншими абонентами даного провайдеру.

На рисунку 3.5 показаний приклад використання орендованих каналів для побудови корпоративної мережі підприємства із трьома філіями.

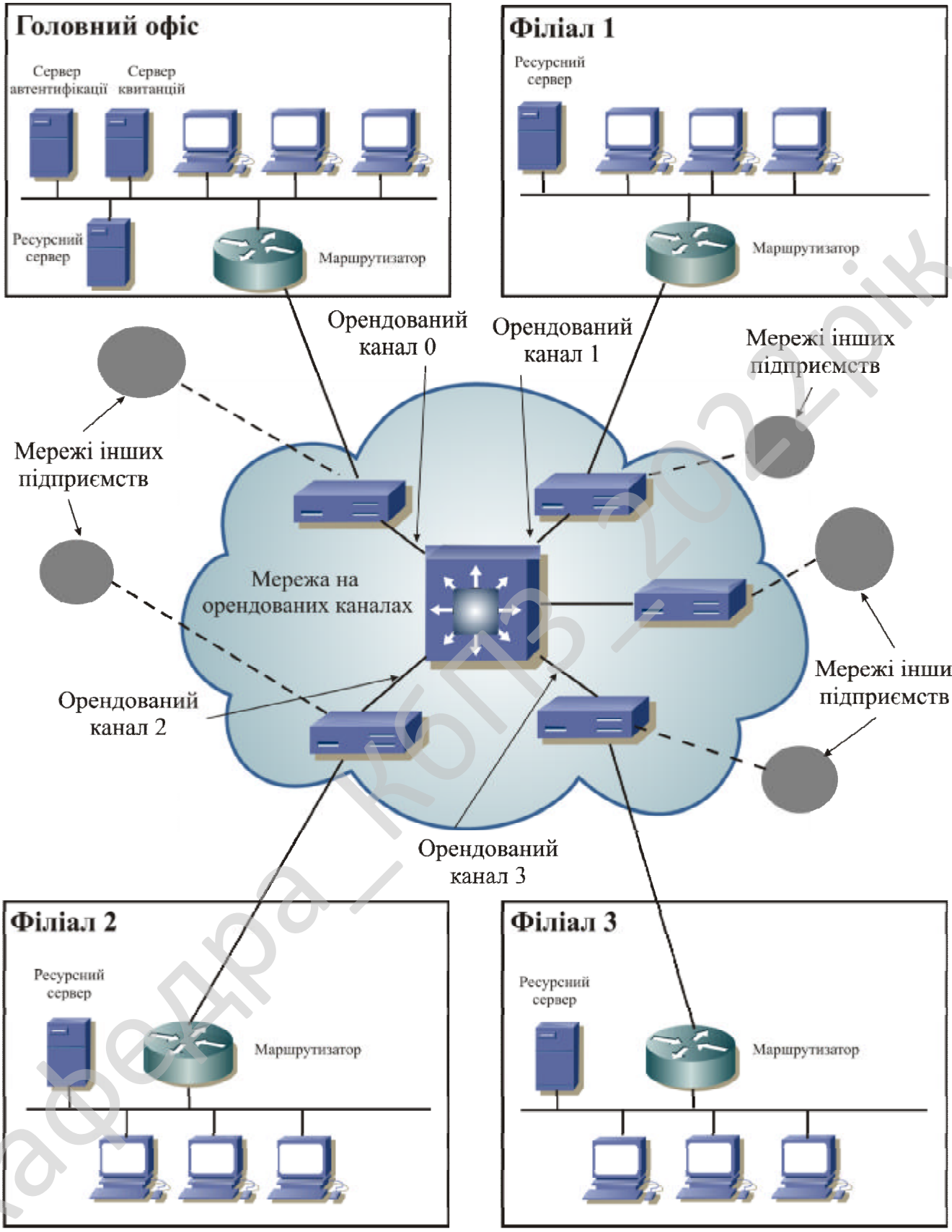

Рисунок 3.5 – Структурна схема розробленої системи

Канали, що зв'язують центральну мережу підприємства з мережами філій, проходять через мультиплексор, що поєднує канали всіх абонентів у магістральний канал. Незважаючи на те, що територіальні канали в цьому випадку не відносяться до власності підприємства, корпоративні мережі, побудовані на орендованих каналах, також називають часними, принаймні по двох причинах. По-перше, смуга пропускання орендованого каналу повністю виділяється підприємству, і тому є в деякому змісті його "приватною власністю". Це повною мірою відноситься до орендованих цифрових каналів, які підтримуються провайдером на базі первинної цифрової мережі з технікою мультиплексування TDM. Орендар такого каналу одержує у своє повне розпорядження всю його пропускну здатність – 64 Кбіт/с, 128 Кбіт/с, 2 Мбіт/с, або вище. У застарілих мережах із частотним мультиплексуванням FDM орендар самостійно розпоряджається не пропускною здатністю, а заздалегідь відомою смугою пропущення каналу. У кожному разі, пропускну здатність каналу підприємство-орендар не ділить ні з ким, і це дуже важливо для створення корпоративної мережі зі стабільними характеристиками.

Наявність гарантованої пропускної здатності дає можливість адміністраторові мережі планувати роботу додатків через глобальні канали зв'язку: розподіляти пропускну здатність каналу між додатками, оцінювати можливі затримки повідомлень, обмежувати обсяг генеруємого територіального трафіку, визначати максимальну кількість активних додатків і т.п.

По-друге, приватний характер мереж, побудованих на орендованих каналах, підтверджується достатньою конфіденційністю даних. Корпоративні дані практично не доступні для абонентів, що не є користувачами корпоративної мережі або співробітниками організації-провайдеру каналів. Дійсно комутацію каналів у первинних мережах може виконати тільки оператор мережі, а рядовому користувачеві така операція недоступна. Це спричиняє більший ступінь захищеності даних, переданих по каналах первинних мереж. Наприклад, тут неможлива типова для Інтернету атака – відгалуження й аналіз "чужого" трафіка

іншим користувачем. Таким чином забезпечується прийнятна безпека переданих даних від зовнішніх атак.

Для мережі центрального офісу найкраще підійдуть маршрутизатори Cisco 3620 або Cisco 3640. Конкретна модель маршрутизатора й кількість установлених модулів буде залежати від покладеного на маршрутизатор завдання.

Моделі серії 3600 надають функціонально повне рішення для організації віддаленого доступу. Інакше кажучи, дані пристрої можуть використовуватися як потужний сервер доступу. Будь-яка модель Cisco 3600 може забезпечувати надійний доступ до Вашої WAN численних віддалених і мобільних користувачів. При цьому вони зможуть не тільки працювати з файловим господарством WAN, як зі своїм власним, але також використовувати загальні програмні додатки.

Дана серія маршрутизаторів є відмінними засобом вкладення коштів, при якому Ви зможете легко в майбутньому змінювати конфігурацію Вашої глобальної мережі в міру росту вимог до кількості з'єднань або в міру появи нових технологій для глобальних обчислювальних мереж. Все це забезпечує захист інвестицій у встаткування й, отже, економію грошей. Моделі Cisco 3600 мають достатню масштабованість. Наприклад, модель Cisco 3640 підтримує інтерфейси ISDN PRI (Primary Rate Interface) або ISDN BRI (Basic Rate Interface) у тому самому шасі: максимальне число підтримуваних PRI-з'єднань – шість, BRI-зєднань – 24, а модель Cisco 3620 може бути сконфігурована з одним портом Ethernet і одним портом ISDN PRI, або одним портом Ethernet і чотирма портами ISDN BRI.

В усі моделі сімейства Cisco 3600 інтегрована міжмережева операційна система Cisco IOS, що підтримує встановлення з'єднань на вимогу, що забезпечує об'єднання локальних мереж, безпека доступу й даних і оптимізацію з'єднань із глобальними обчислювальними мережами. Завдяки підтримці повного набору функціональних можливостей Cisco IOS, маршрутизатори серії Cisco 3600 дають надійні й гнучкі засоби роботи з Інтернетом/Інтранетом мультимедійними додатками. Cisco IOS робить легкою не тільки роботу через Інтернет, але спрощує

й підвищує ефективність роботи у корпоративній мережі Інтернет.

Маршрутизатори сімейства Cisco 3600 надають відмінну можливість вибору конфігурації. Так, наприклад, модель Cisco 3640 має чотири слота під мережеві модулі, а модель Cisco 3620 – два слоти. У кожний слот по вибору можуть бути встановлені мережеві модулі для Інтернет і Token Ring, а також можна вибирати із цілого набору інтерфейсних карт для з'єднання із глобальними обчислювальними мережами. Кожне рознімання для мережевих модулів може прийняти інтерфесні карти мережевих модулів різного типу, включаючи ISDN PRI, ISDN BRI, і асинхронні/синхронні послідовні інтерфейси.

Інакше кажучи, по наявній лінії зв'язку можна легко об'єднати за допомогою Cisco 3640 WAN із чотирма іншими локальними мережами, розташованими в інших кінцях міста, і працювати в них, як у власній мережі й навіть використовувати їхнє устаткування (наприклад, високоякісний принтер).

Сімейство Cisco 3600 надає користувачеві можливість ефективного й недорогого використання додатків Інтернету/Інтранету і можливість масштабування подібних рішень при збільшенні потреб або зміні структури віддаленого доступу.

Пропонуючи повну інтеграцію мережевого маршрутизатора й сервера для ISDN, асинхронних і синхронних з'єднань WAN в одному продукті, Cisco 3600 дає користувачеві нову платформу для майбутніх додатків, що є негайним рішенням сьогоднішніх проблемам.

Для віддалених офісів (філій) найкраще підійдуть маршрутизатори серій Cisco 2500 або Cisco 2600. Конкретна модель маршрутизатора й кількість встановлених модулів буде залежати від покладеного на маршрутизатор завдання.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи показана на рисунку 3.6.

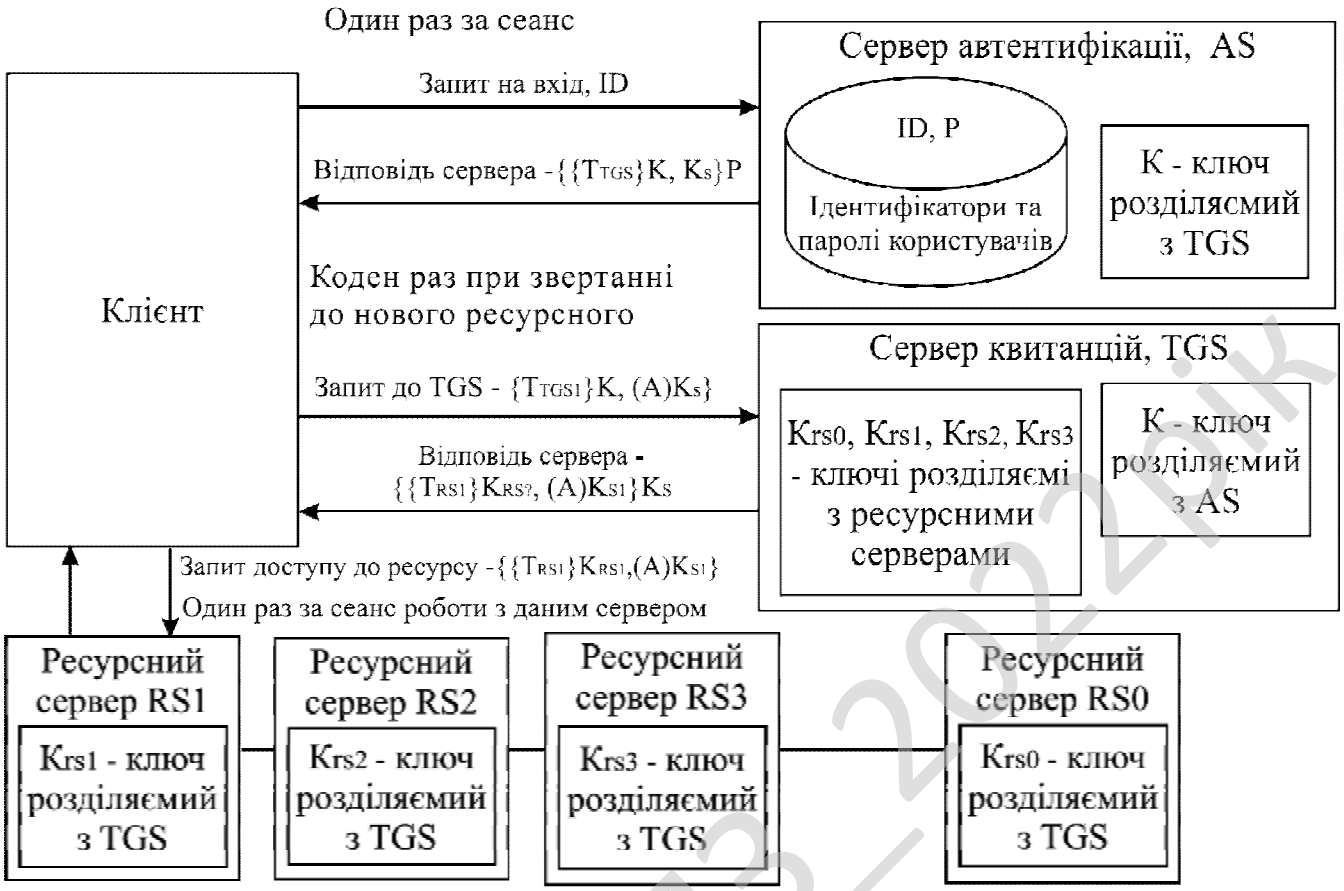

Рисунок 3.6 – Функціональна схема розробленої системи

Розподіл ключів та отримання за допомогою них доступу до ресурсів мережі WAN проходить у декілька етапів:

1. Первинна автентифікація.
2. Одержання дозволу на доступ до ресурсного сервера.
3. Одержання доступу до ресурсу.

Первинна автентифікація

Процес доступу користувача до ресурсів включає дві процедури: по-перше, користувач повинен довести свою легальність (автентифікація), а по-друге, він повинен одержати дозвіл на виконання певних операцій з певним ресурсом (авторизація). У розробленій системі користувач один раз автентифікується під час логічного входу в мережу, а потім проходить процедури автентифікації й авторизації щоразу, коли йому потрібен доступ до нового ресурсного сервера.

Виконуючи логічний вхід у мережу, користувач, а точніше клієнтська програма, установлена на його комп'ютері, посилає автентифікаційному серверу AS ідентифікатор користувача ID (рисунок 3.6).

Спочатку автентифікаційний сервер перевіряє в базі даних, чи є запис про користувача з таким ідентифікатором, потім, якщо такий запис існує, витягає з неї пароль користувача p. Даний пароль буде потрібний для шифрування всієї інформації, що направить автентифікаційний сервер клієнтові як відповідь. А відповідь складається із квитанції T_{TGS} на доступ до сервера квитанцій Kerberos і ключа сеансу K_s. Під сеансом тут розуміється увесь час роботи користувача, від моменту логічного входу в мережу до моменту логічного виходу. Ключ сеансу буде потрібний для шифрування в процедурах автентифікації протягом усього користувальницького сеансу. Квитанція шифрується за допомогою секретного DES-ключа K, який розділяють автентифікаційний сервер і сервер квитанцій. Все разом – зашифрована квитанція й ключ сеансу – ще раз шифруються за допомогою користувальницького пароля p. Таким чином, квитанція шифрується двічі ключем K и паролем p. У наведених вище позначеннях повідомлення-відповідь, що автентифікаційний сервер посилає клієнтові, виглядає так: $\{\{T_{TGS}\}K, K_s\}p$.

Після того як таке відповідне повідомлення надходить на клієнтську машину, клієнтська програма просить користувача ввести свій пароль. Коли користувач вводить пароль, то клієнтська програма намагається за допомогою пароля розшифрувати повідомлення, що надійшло. Якщо пароль вірний, то з повідомлення витягається квитанція на доступ до сервера квитанцій $\{T_{TGS}s\}K$ (у зашифрованому виді) і ключ сеансу K_s (у відкритому виді). Успішна розшифровка повідомлення означає успішну автентифікацію. Помітимо, що автентифікаційний сервер AS автентифікує користувача без передачі пароля по мережі.

Квитанція T_{TGS} на доступ до сервера квитанцій TGS є посвідченням легальності користувача й дозволом йому продовжувати процес одержання доступу до ресурсу. Ця квитанція містить:

– ідентифікатор користувача;

– ідентифікатор сервера квитанцій, на доступ до якого отримана квитанція;

– оцінку про поточний час;

– період часу, протягом якого може тривати сеанс;

– копію ключа сесії KS.

Як уже було сказано, клієнт має квитанцію в зашифрованому виді. Шифрування підвищує впевненість у тому, що ніхто, навіть сам клієнт – власник даної квитанції, – не зможе квитанцію підробити, підмінити або змінити. Тільки сервер TGS, одержавши від клієнта квитанцію, зможе її розшифрувати, тому що в його розпорядженні є ключ шифрування К.

Час дії квитанції обмежений тривалістю сеансу. Дозволена тривалість сеансу користувача, що втримується у квитанції на доступ до сервера квитанцій, задається адміністратором і може змінюватися залежно від вимог до захищеності мережі. У мережах із твердими вимогами до безпеки час сеансу може бути обмежено 30 хвилинами, в інших умовах цей час може скласти 8 годин. Інформація, що утримується у квитанції, визначає її строк придатності. Надання квитанції на цілком певний час захищає її від неавторизованого користувача, що міг би її перехопити й використовувати в майбутньому.

Одержання дозволу на доступ до ресурсного сервера

Наступним етапом для користувача є одержання дозволу на доступ до ресурсного сервера (наприклад, до файлового сервера або сервера додатків). Але для цього треба звернутися до сервера TGS, що видає такі дозволи (квитанції). Щоб одержати доступ до сервера квитанцій, користувач уже отримав квитанцією $\{T_{TGS}\}K$, видану йому сервером AS. Незважаючи на захист паролем і шифрування, користувачу, для того щоб довести серверу квитанцій, що він має право на доступ до ресурсів мережі, потрібно ще дещо, крім квитанції.

Як уже згадувалося, перше повідомлення від автентифікаційного сервера містило не тільки квитанцію, але й секретний ключ сеансу K_s, що розділяється із

сервером квитанцій TGS. Клієнт використовує цей ключ для шифрування ще однієї електронної форми, що називається автентифікатором {A}K$. Автентифікатор A містить ідентифікатор і мережеву адреса користувача, а також власну часову відмітку. На відміну від квитанції $\{T_{TGS}\}K$, що протягом сеансу використовується багаторазово, автентифікатор призначений для одноразового використання й має дуже короткий час життя – звичайно кілька хвилин. Клієнт посилає серверу квитанцій TGS повідомлення-запит, що містить квитанцію й автентифікатор: $\{T_{TGS}\}K$, $\{A\}K_S$.

Сервер квитанцій розшифровує квитанцію наявним у нього ключем K, перевіряє термін дії квитанції, і витягає з неї ідентифікатор користувача.

Потім сервер TGS розшифровує автентифікатор, використовуючи ключ сеансу користувача K_s, який він витягає із квитанції. Сервер квитанцій порівнює ідентифікатор користувача і його мережеву адресу з аналогічними параметрами у квитанції й повідомленні. Якщо вони збігаються, то сервер квитанцій одержує впевненість, що дана квитанція дійсно представлена її законним власником.

Помітимо, що звичайне володіння квитанцією на одержання доступу до сервера квитанцій не доводить ідентичність користувача. Тому що автентифікатор дійсний тільки протягом короткого проміжку часу, то малоймовірно вкрасти одночасно й квитанцію, і автентифікатор і використовувати їх протягом цього часу. Щоразу, коли користувач звертається до сервера квитанцій для одержання нової квитанції на доступ до ресурсу, він посилає багаторазово використовувану квитанцію й новий автентифікатор.

Клієнт звертається до сервера квитанцій за дозволом на доступ до ресурсного сервера, що тут позначений як RS1. Сервер квитанцій, упевнившись у легальності запиту й особистості користувача, відсилає йому відповідь, що містить дві електронні форми: багаторазово використовувану квитанцію на одержання доступу до запитуваного ресурсного сервера T_{RS1} і новий ключ сеансу K_{S1}.

Квитанція на одержання доступу шифрується секретним ключем K_{RS1} поділюваним тільки сервером квитанцій і тим сервером, до якого надається

доступ, у цьому випадку – RS1. Сервер квитанцій розділяє унікальні секретні ключі з кожним сервером мережі. Ці ключі розподіляються між серверами мережі фізичним способом або яким-небудь іншим секретним способом при установці системи. Коли сервер квитанцій передає квитанцію на доступ до будь-якого ресурсного сервера, то він шифрує її, так що тільки цей сервер зможе розшифрувати її за допомогою свого унікального ключа.

Новий ключ сеансу K_{S1} утримується не тільки в самому повідомленні, що посилається клієнтові, але й усередині квитанції T_{RS1}. Все повідомлення шифрується старим ключем сеансу клієнта K_s, так що його може прочитати тільки цей клієнт. Використовуючи введені позначення, відповідь сервера TGS клієнтові можна представити в наступному виді: $\{\{T_{RS1}\}K_{RS1}, K_{S1}\}K_S$.

Одержання доступу до ресурсу

Коли клієнт розшифровує повідомлення, що надійшло, то він відсилає серверу, до якого він хоче одержати доступ, запит, що містить квитанцію на одержання доступу й автентифікатор, зашифрований новим ключем сеансу: $\{TR_{si}\}KR_{si}$, {A}K$1.

Це повідомлення обробляється аналогічно тому, як оброблявся запит клієнта сервером TGS. Спочатку розшифровується квитанція ключем K_{RS1}, потім витягається ключ сеансу K_{si} і розшифровується автентифікатор. Далі порівнюються дані про користувача, що утримуються у квитанції й автентифікаторі. Якщо перевірка проходить успішно, то доступ до мережевого ресурсу дозволений.

На цьому етапі клієнт також може захотіти перевірити автентичність сервера перед тим, як почати з ним працювати. Взаємна процедура автентифікації запобігає будь-якій можливості спроби одержання неавторизованим користувачем доступу до секретної інформації від клієнта шляхом підміни сервера.

Автентифікація ресурсного сервера в розробленій системі виконується у відповідності з наступною процедурою. Клієнт звертається до сервера із пропозицією, щоб той надіслав йому повідомлення, у якому повторив часову мітку з автентифікатора клієнта, збільшену на 1. Крім того, потрібно, щоб дане

повідомлення було зашифровано ключем сеансу K_{si}. Щоб виконати такий запит клієнта, сервер витягає копію ключа сеансу із квитанції на доступ, використовує цей ключ для розшифровки автентифікатора, нарощує значення часової мітки на 1, заново зашифровує повідомлення, використовуючи ключ сеансу, і повертає повідомлення клієнтові. Клієнт розшифровує це повідомлення, щоб одержати збільшену на одиницю оцінку часу.

При успішному завершенні описаного процесу клієнт і сервер впевнюються у таємності своїх транзакцій. Крім цього, вони одержують ключ сеансу, який можуть використовувати для шифрування майбутніх повідомлень.

3.4 Розробка діаграми процесів

Розглянемо діаграму процесів розробленої системи, що зображена на рисунку 3.7.

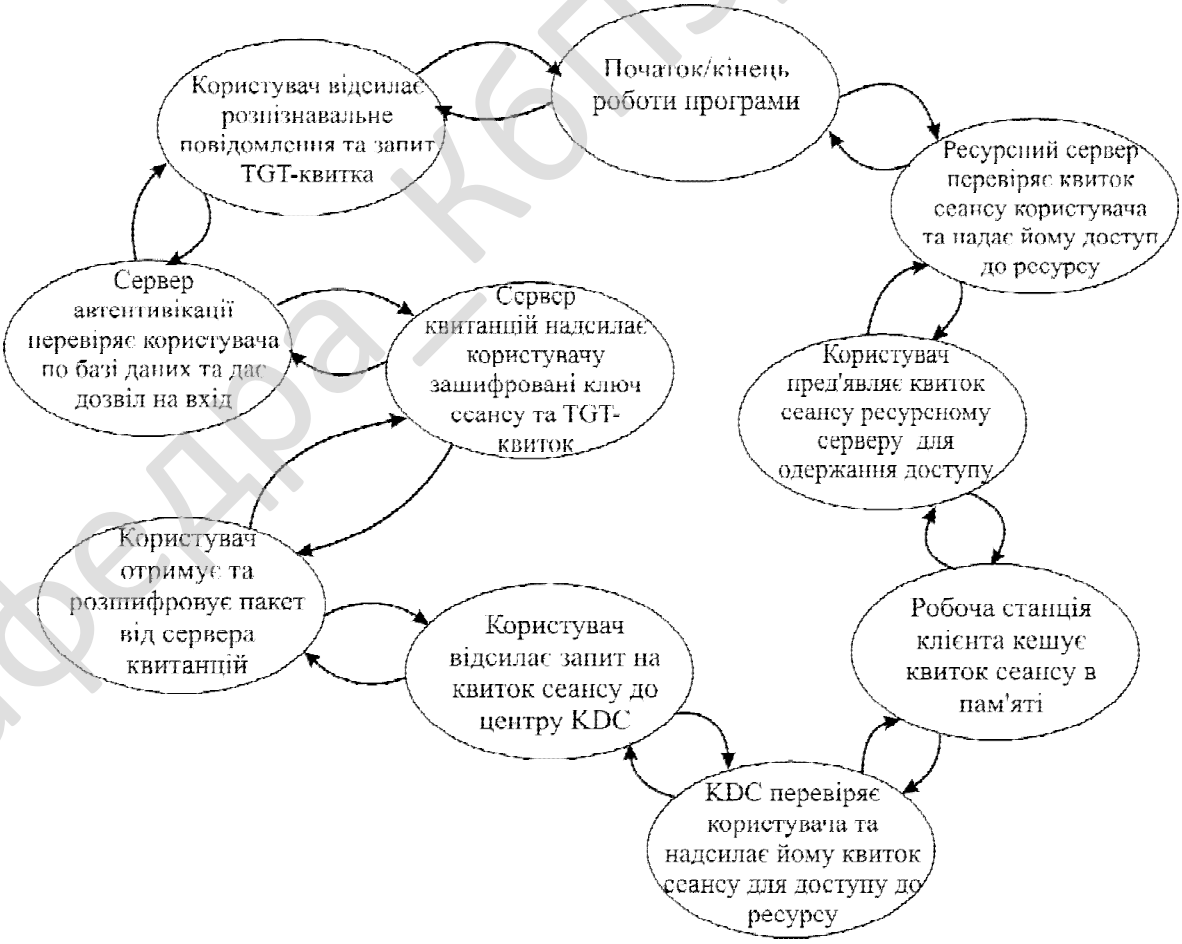

Рисунок 3.7 – Діаграма процесів розробленої системи

На діаграмі процесів відображено, що коли користувач входить у систему, вдруковуючи ім'я користувача й пароль, комп'ютер клієнта застосовує однобічне гешування до пароля користувача для створення секретного ключа, що кешується в надійній пам'яті на комп'ютері. Однобічне гешування означає, що пароль не може бути відновлений виходячи з геш-значення (hash).

Для здійснення процесу входу клієнта в систему клієнт і сервер виконують наступні дії:

1. Користувач з робочої станції посилає розпізнавальне повідомлення Центру розподілу ключів (KDC – Key Distribution Center). Це повідомлення включає:

– ім'я користувача;

– область (realm) користувача (ім'я домена);

– запит на TGT-квиток;

– попередні розпізнавальні дані, які включають мітку часу.

Попередні розпізнавальні дані зашифровані за допомогою секретного ключа, отриманого з користувальницького пароля.

2. Коли повідомлення досягає сервера, сервер досліджує ім'я користувача, а потім перевіряє базу даних каталогу в пошуках своєї копії секретного ключа, пов'язаного з даним обліковим записом користувача. Сервер розшифровує зашифровані в повідомленні дані за допомогою секретного ключа й перевіряє часову мітку. Якщо розшифровка пройшла успішно, і часова мітка відрізняється від поточного часу на сервері в межах 5 хвилин, сервер готовий підтвердити дійсність користувача. Якщо розшифровка виявиться невдалою, це означає, що користувач ввів неправильний пароль, і автентифікація зазнає невдачі. Якщо часова мітка відрізняється більше ніж на 5 хвилин від поточного часу на сервері, то автентифікація також зазнає невдачі. Причина такої маленької різниці в часі полягає в тому, що вона повинна запобігти можливій спробі перехоплення розпізнавального пакета з наступним повторенням його пізніше. Задана за замовчуванням максимально припустима різниця в часі, що становить 5 хвилин,

може бути сконфігурована в політику захисту домену.

3. Після автентифікації користувача сервер посилає клієнтові повідомлення, що включає ключ сеансу й TGT. Ключ сеансу – це ключ шифрування, що клієнт буде використовувати для взаємодії з KDC замість секретного ключа клієнта. TGT – це квиток сеансу, що надає користувачу доступ до контролера домену. Протягом терміну служби TGT клієнт пред'являє TGT контролеру домену щоразу, коли йому потрібно звернутися до мережевих ресурсів. Повне повідомлення від сервера зашифроване за допомогою секретного ключа користувача. Крім того, квиток TGT зашифрований за допомогою довгострокового секретного ключа сервера.

4. Коли пакет прибуває на комп'ютер клієнта, секретний ключ користувача використовується для розшифровки пакета. Якщо розшифровка пройшла успішно й часова мітка припустима, то комп'ютер користувача припускає, що центр KDC надійно ідентифікував користувача, тому що йому знайомий його секретний ключ. Ключ сеансу потім кешується на локальному комп'ютері, поки не скінчиться строк його дії або поки користувач не зробить вихід із системи робочої станції. Цей ключ сеансу буде використовуватися для шифрування всіх майбутніх підключень до центра KDC, тобто клієнт більше не повинен пам'ятати секретний ключ, і він видаляється з кешу робочої станції. Квиток TGT зберігається в зашифрованій формі в кеші робочої станції.

5. Користувач опізнаний, але він усе ще не має ніякого доступу до мережевих ресурсів. TGT – це квиток сеансу, що надає доступ до Центру розподілу ключів, але щоб одержати доступ до будь-яких інших мережевих ресурсів, користувач повинен одержати інший квиток сеансу від KDC. Робоча станція клієнта надсилає запит на квиток сеансу до центру KDC. Запит включає ім'я користувача, квиток TGT, наданий у процесі автентифікації, ім'я мережевої служби, до якої користувач хоче одержати доступ, і часову мітку, що зашифрована з використанням ключа сеансу, отриманого в процесі AS Exchange.

6. Служба KDC розшифровує квиток TGT, використовуючи свій

довгостроковий ключ. Потім вона витягає ключ сеансу із квитка TGT і розшифровує часову мітку, щоб переконатися, що клієнт використовує правильний ключ сеансу, і гарантувати, що часова мітка припустима. Якщо ключ сеансу й часова мітка прийнятні, то KDC готовить квиток сеансу для доступу до мережевої служби.

7. Квиток сеансу включає дві копії ключа сеансу, що клієнт буде використовувати для з'єднання з необхідним ресурсом. Перша копія ключа сеансу зашифрована, використовуючи ключ сеансу клієнта, отриманий у процесі початкового входу в систему. Друга копія ключа сеансу призначена для мережевої служби й включає інформацію про доступ користувача. Ця частина квитка сеансу зашифрована, використовуючи секретний ключ мережевої служби, що невідомий робочої станції клієнта, але відомий і службі KDC і мережевій службі, тому що сервер, на якому розташований ресурс, є членом сфери KDC.

8. Робоча станція клієнта кешує обидві частини квитка сеансу в пам'яті.

9. Тепер клієнт пред'являє квиток сеансу мережевій службі для одержання доступу.

10. Мережева служба розшифровує ключ сеансу, зашифрований у квитку сеансу, використовуючи довгостроковий ключ, яким вона володіє разом із центром KDC. Якщо ця розшифровка пройшла успішно, то мережева служба знає, що квиток виданий довіреною службою KDC. Потім мережева служба розшифровує лексему доступу користувача, використовуючи ключ сеансу, і перевіряє користувальницький рівень доступу. Запит клієнта включає також часову мітку, що зашифрована за допомогою ключа сеансу й перевірена сервером.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми, блок-схема якого зображена на рисунку 4.1.

З рисунку видно, що після запуску програми на екрані з'являється вікно авторизації. Користувач повинен ввести логін та пароль і натиснути кнопку "Вхід". Після цього програма перевіряє наявність введеного логіну у базі даних користувачів. Якщо такий логін є, то перевіряється правильність введення паролю. При виявлені помилки у вікні виводиться фраза "Помилка авторизації" та очікується нове введення логіну та паролю. Якщо авторизація пройшла успішно, то програма перевіряє, які права має авторизований користувач. В розробленій системі є три типи облікових записів:

1. Адміністратор.
2. Користувач.
3. Обліковий запис, строк дії якого закінчився.

Якщо обліковий запис має права адміністратора, то відкривається вікно програми-сервера. Якщо ж обліковий запис належить користувачу, то відкривається вікно програми-клієнта.

Якщо обліковий запис закінчив строк дії, а це може бути у випадку коли він належить звільненому працівнику, працівнику тимчасово позбавленому прав доступу до корпоративної мережі або в наслідок помилки адміністратора, чи збою системи, то на екран виводиться повідомлення "Строк дії даного облікового запису завершився, зверніться до адміністратора системи."

Як видно з вище сказаного програма має два режими роботи: робота в режимі адміністратора (серверний додаток), робота в режимі користувача (клієнтський додаток). По завершенню роботи можна змінити режим, або вийти з програми.

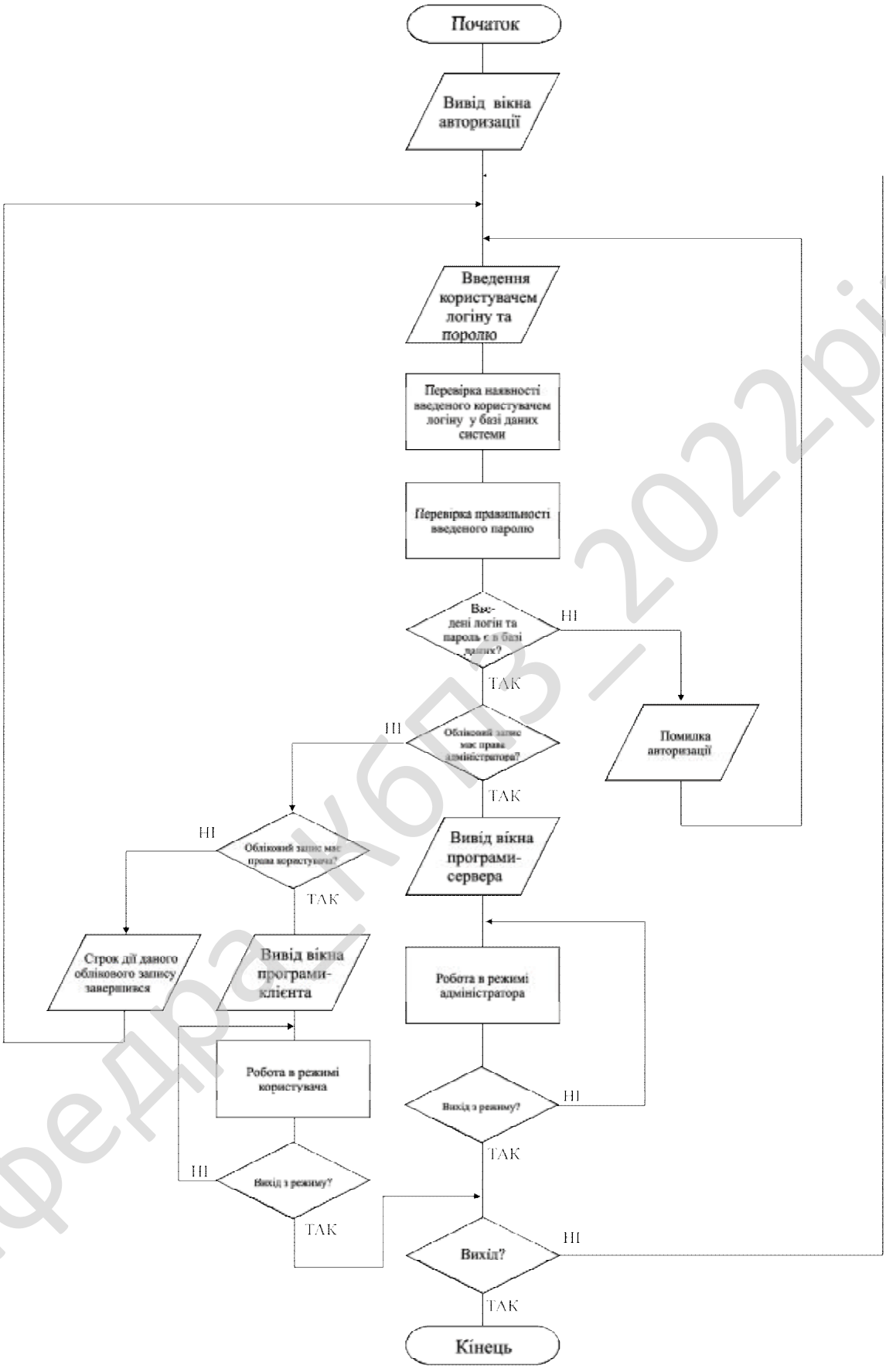

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

Алгоритм роботи клієнтського додатку зображень на рисунку 4.2, серверного – на рисунку 4.3.

Розроблена система забезпечує механізм взаємної автентифікації між клієнтом і сервером у мережі, у якій пакети, що передаються, можуть бути переглянуті або модифіковані.

Щоб належним чином забезпечити взаємну автентифікацію використовуються симетричні ключі, шифровані об'єкти й служби. Цей підрозділ містить наступні розділи:

- Використання паролів та ключів при автентифікації.
- TGT-квитки.
- Сеансові ключі.
- Сеансові квитки.
- Центр розподілу ключів.
- Область Kerberos.

Концепція поділюваного секрету проста: якщо секрет відомий тільки двом людям, тоді кожна людина зможе перевірити дійсність іншого, упевнившись, що інша людина знає секрет.

Розроблений протокол вирішує проблему з безпечним ключем криптографічно. Замість доступного пароля партнери по з'єднанню створюють криптографічний ключ і використовують знання про цей ключ, щоб перевірити дійсність один одного. Додатково, щоб метод працював, відкриті ключі повинні бути симетричними, тобто один ключ повинен використовуватися й для шифрування й для дешифрування. Одна сторона підтверджує знання про ключ шляхом шифрування частини інформації, інша засвідчує знання про ключ дешифруванням інформації.

Використовується декілька ключів і типів ключів для шифрування. Типи ключів можуть включати довгострокові симетричні ключі, довгострокові асиметричні ключі й короткострокові симетричні ключі. Протокол автентифікації призначений для використання симетричного шифрування, це означає, що один

відкритий ключ використовується відправником і одержувачем для шифрування й дешифрування.

Довгостроковий симетричний ключ: користувач, система, служба, і міжобласні ключі

Довгостроковий симетричний ключ створюється з пароля. Відкритий текст пароля трансформується в криптографічний ключ передачею тексту пароля через криптографічну функцію (всі реалізації протоколу автентифікації Kerberos повинні підтримувати DES-CBC-MD5, інші алгоритми допускаються). Результат криптографічної функції – це ключ.

Користувальницькі ключі

Коли створюється користувач, пароль використовується для створення ключа користувача. У доменах Active Directory ключ користувача зберігається в об'єкті користувача в Active Directory. На робочій станції користувальницький ключ створюється коли користувач входить в ОС.

Системні ключі

Коли робоча станція або сервер входять у домен Windows, вони одержують пароль. Таким же способом, як і у випадку користувальницького акаунта пароль системного акаунта використовується для створення системного ключа.

Ключі служб

Служби використовують ключ, заснований на паролі акаунта, коли вони запускаються. Всі KDC в одній області використовують однаковий ключ служби. Ці ключі засновані на паролі, призначеному акаунту krbtgt. Кожний домен Active Directory має цей вбудований акаунт.

Міжобласні ключі

Додатково до крос-обласної автентифікації KDC повинні створюватися внутрішньообласні ключі. Області повинні довіряти один одному тому що вони мають загальний ключ. Домени Active Directory, що мають батьківські відносини розділяють внутрішньообласний ключ. Цей внутрішньообласний ключ,

заснований на відносинах “transitive trusts” в Windows 2000 і Windows Server 2003. Якщо відносини довіри створені, два домени можуть обмінюватися ключами специфічними для їхньої довіри.

Довжини ключів шифрування

Програма підтримує різні типи шифрування й довжини ключів залежно від виконуваного завдання й певних опцій. Незважаючи на те що розмір ключа визначає рівень захисту забезпечуваний ключем, розмір ключа дещо впливає на розмір квитка.

Наступна таблиця перераховує довжини ключів, що підтримують різні типи шифрування.

Таблиця 4.1 – Довжини ключів для різних типів шифрування, що підтримуються розробленою програмою

Алгоритм шифрування	Довжина ключа
RC4-HMAC	128
DES-CBC-CRC	56
DES-CBC-MD5	56

Довгострокові асиметричні ключі – відкритий ключ

Сертифікати з відкритим ключем, що зберегаються на смарт-картах, мають тільки довгострокові асиметричні ключі реалізації Microsoft для автентифікації Kerberos.

Короткострокові симетричні ключі – ключі сесії

Сесійні ключі, що використовуються для квитка на одержання TGT-квитків і квитків сесії на одержання сервісів тимчасово використовуються настільки довго, наскільки сесія або служба працездатні.

Квитки

Основний компонент автентифікації типу Kerberos – це квиток. Повідомлення Kerberos використовуються для запиту або випуску квитків. Є два типи квитків – (TGT) і квитки сесії на одержання сервісу (TGS).

Запити квитків

Клієнти посилають запити квитків до Центру сертифікатів.

Таблиця 4.2 – Типи запитів квитків

Тип запитаного квитка	Служба KDC, що надсилає запит
TGT-квиток	Служба автентифікації
Квиток сесії	Служба видачі квитків

Запит квитка включає:

– Властивості запиту (прапори), такі як період перевідновлення квитка.

– Метод шифрування запиту.

Квиток на одержання квитків

KDC відповідає запиту служби автентифікації клієнта поверненням квитка для себе. Це особливий квиток служби, що називається квитком на одержання квитків (TGT). TGT дозволяє службі автентифікації безпечно передавати посвідчення користувача службі видачі квитків.

TGT це:

– Початковий квиток користувача служби автентифікації.

– Використовується, щоб запросити квиток на отримання сервісу.

– Означає тільки використання службою видачі квитків.

TGT шифрується ключем спільним з KDC. Клієнт не може прочитати квиток TGT. Тільки сервер KDC може прочитати TGT для безпечного доступу до користувальницького посвідчення, сесійним ключам та іншій інформації. Подібно звичайному квитку служби TGT містить копію сесійного ключа, що служба (у даному випадку KDC) використовує для зв'язку із клієнтом. TGT зашифрований довгостроковим ключем KDC.

З погляду клієнта TGT це інший квиток. Перед тим як він намагається приєднатися до будь-якого сервісу, клієнт спочатку перевіряє кеш посвідчень на наявність квитка на одержання сервису (TGS) для цього сервісу. Якщо його немає клієнт перевіряє кеш знову на наявність TGT. Якщо він знаходить TGT то дістає

відповідний TGS сесійний ключ із кешу, і використовує цей ключ для підготовки автентифікатору та посилає автентифікатор і TGT в KDC.

З боку KDC TGT дозволяє уникнути виконання зайвих дій по пошуку користувальницького довгострокового ключа щоразу, коли користувач запитує сервіс. KDC дивиться користувальницький довгостроковий ключ один раз, коли він надає початковий TGT.

Для всіх обмінів із цим клієнтом KDC може дешифрувати TGT його власним довгостроковим ключем, витягаючи сесійний ключ і використовувати його для перевірки автентифікатору клієнта.

Квитки на одержання сервісу

Квиток на одержання сервісу TGS дозволяє безпечно передавати посвідчення особи запитуючого до цільового на серверу або службі. KDC відповідає на запит клієнта про доступ до служби, висилаючи 2 копії сесійного ключа клієнтові. Клієнтська копія сесійного ключа зашифрована ключем, що KDC розділяє із клієнтом.

Копія сесійного ключа для одержання сервісу вкладена разом з інформацією про клієнта в структуру даних, що називається квитком на одержання сервісу. Вся структура зашифрована ключем, що KDC розділяє із сервером, який надає сервіс. Обов'язок клієнта тепер управляти контактом із сервером, використовуючи квиток із сесійним ключем на доступ до сервісу. Квиток на одержання сервісу використовується для автентифікації службою й називається цільовою службою.

Квиток на одержання сервісу шифрується сесійним ключем, що є довгостроковим ключем поділюваним KDC і цільовою службою. Таким чином, незважаючи на те, що клієнт управляє квитком на одержання сервісу клієнт не може прочитати його. Тільки KDC і цільова служба можуть читати квиток, одержуючи безпечний доступ до користувальницького посвідчення, сесійного ключа й іншої інформації.

Коли клієнт одержує відповідь KDC до сервера він витягає квиток і

клієнтську копію сесійного ключа, розміщює їх у безпечний кеш (розташований в оперативній пам'яті, не на диску) Коли клієнт одержує відповідь KDC він витягає квиток і клієнтську копію сесійного ключа поміщає обидва у безпечний кеш розташований у пам'яті. Коли клієнт хоче одержати доступ до сервера він висилає серверу повідомлення, що складається із квитка, що усе ще зашифрований секретним ключем сервера, автентифікатор, що зашифрований сесійним ключем. Квиток і автентифікатор разом становлять клієнтське посвідчення до сервера.

Переваги квитків на одержання сервісу

Сервер не зберігає сесійний ключ, що використовується для зв'язку із клієнтом. Це відповідальність клієнта зберігати квиток сервера в кеші для посвідчень і надавати квиток щоразу, коли йому потрібний доступ до сервера. Щоразу, коли сервер одержує квиток на одержання сервісу від клієнта, сервер може використовувати закритий ключ для розшифрування квитка й добування сесійного ключа. Коли серверу не потрібний сесійний ключ він може видалити його.

Клієнт не має потреби в тому, щоб звертатися до KDC щоразу, коли він хоче одержати доступ до цільового сервера. Для захисту від можливості крадіжки копії квитка квиток на одержання сервісу має час дії, що визначає KDC у структурі даних квитка. Як довго квиток діє залежить від політики області. Коли користувач виходить із мережі кеш посвідчень очищується й всі квитки на одержання сервісу також як і всі сесійні ключі знищуються.

Структура квитка

Точні структури даних для квитка як повідомлення, можуть бути знайдені в RFC 1510. Перші три поля у квитку не шифруються. Інформація – відкритий текст, для того щоб клієнт міг використовувати інформацію для керування квитками в його кеші. Зміст квитка наведений у таблиці 4.3.

Таблиця 4.3 – Зміст квитка

Ім'я поля	Опис
Номер версії квитка	5
Область	Ім'я області (домену), що випускає квиток. KDC може випускати квитки тільки для серверів у власній області.
Ім'я сервера	Ім'я сервера
Наступні поля шифруються закритим ключем сервера	
Прапори	Опції які визначають, коли квиток може використовуватися.
Ключ	Сесійний ключ для шифрування й дешифрування клієнтських повідомлень і повідомлень цільового сервера.
Клієнтська область	Ім'я області, яку запросили
Ім'я клієнта	Ім'я користувача, що надіслав запит
Передача	Список областей Kerberos, які є частиною процесу автентифікації клієнта під час крос-обласної автентифікації.
Час автентифікації	Час початкової автентифікації клієнта. KDC розташовує часову мітку цьому полі, коли випускає TGT. Коли KDC випускає квитки, засновані на TGT, він копіює час автентифікації TGT у поле часу автентифікації квитка. Додаток повинен мати політику обмеження квитків, що базується на старих TGT, що вимагає від клієнта одержати новий TGT і потім новий квиток на одержання сервісу TGS.

Продовження таблиці 4.3

Ім'я поля	Опис
Час початку	Час, починаючи з якого, квиток придатний.
Час закінчення	Час закінчення дії квитка
Переобновити до	(Опціонально). Максимальний час закінчення дії, що може бути встановлений у квитку в прапорі квитка RENEWABLE
Адреса клієнта	(Опціонально). Одна або більше адрес, для яких квиток може бути випущений. Якщо пропущено, квиток може бути випущений для будь-якої адреси
Дані авторизації	(Опціонально). Це поле містить авторизаційні дані клієнта. У реалізації MIT протоколу Kerberos це поле містить обмеження доступу. Наприклад, якщо цей квиток для сервера друку посилається клієнтом, що бажає роздрукувати файл, ім'я файлу включається тут, обмеження на доступ до принт-сервера, що дозволить клієнтові роздрукувати тільки цей файл. У реалізації Microsoft це поле має значення, тому що містить користувальницький SID і дані SID групи клієнта, у яку він входить, які використовуються для побудови токену доступу клієнта.

Наступна таблиця містить список і опис полів прапорів протоколу Kerberos. Поля прапорів – опціональні бітові поля й установлюються встановленням відповідного біту в 0 або 1. Незважаючи на те що поля мають довжину 32 біти тільки 11 прапорів цікаві адміністраторові.

Таблиця 4.4 – Прапори квитків

FORWARDABLE	(TGT тільки) Говорить службі видачі квитків, що можна випустити новий TGT, заснований на наданому TGT, з іншою мережевою адресою, узятою з наданого TGT.
FORWARDED	Показує, що TGT може бути завчасним або що квиток був випущений на основі завчасного TGT
PROXIABLE	(TGT тільки). Говорить службі, яка видає квитки, що вона може випускати квитки з адресою, що відрізняється від зазначеного в TGT
PROXY	Показує, що мережева адреса у квитку відрізняється від тої ж адреси в TGT, що використовується для одержання квитка.
MAY-POSTDATE	(TGT тільки). Говорить службі видачі квитків TGS, що квиток може бути датований переднім числом.
POSTDATED	Показує, що квиток може бути датований переднім числом
INVALID	Показує, що квиток непридатний. Квиток, датований переднім числом, має ознаку непридатності доти, поки не наступив час початку дії.
RENEWABLE	Використовується в сполученні з полями "час закінчення" і "переобновити до" як основа для періодичного відновлення квитків KDC з більшим часом життя.
INITIAL	Показує, що квиток був випущений з використанням служби автентифікації не на основі TGT
PRE-AUTHENT	Показує, що клієнт був автентифікований KDC перед тим як квиток був випущений. Цей прапор звичайно показує наявність автентифікатору у квитку. Це може також бути прапором, що є присутнім у посвідченні, коли здійснюється вхід зі смарт-картою.
HW-AUTHENT	Показує, що початкова автентифікація вимагає використання друкованої плати.

Рисунок 4.2 – Блок-схема алгоритму роботи клієнтського додатку

Рисунок 4.3 – Блок-схема алгоритму роботи серверного додатку

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації та хмарних обчислень DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі. Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має зручний та інтуїтивно зрозумілий інтерфейс. Вона складається з двох частин: програми-клієнту та програми-серверу.

Для входу до розробленої системи необхідно пройти процес авторизації (рисунок 5.1).

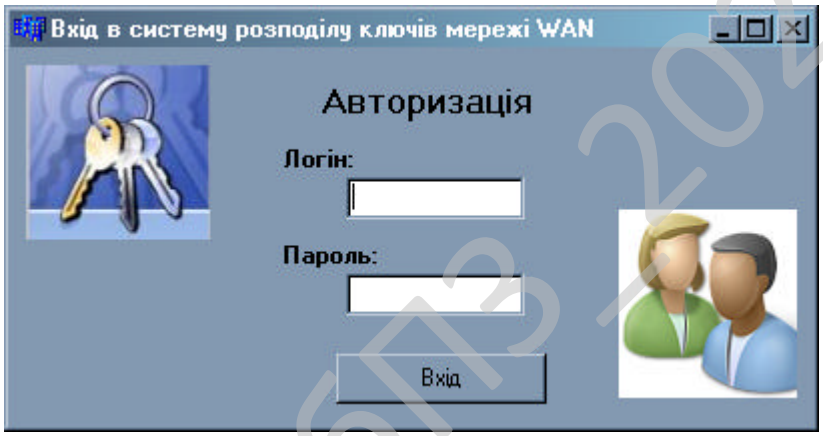

Рисунок 5.1 – Вікно входу в систему розподілу ключів корпоративної мережі WAN

Після введення логіну та паролю відкриється клієнтська або серверна частина програми, в залежності від того, ввійшов в систему користувач, чи адміністратор. Первинний пароль та логін для адміністратора – 2 й 2, для користувача 1 й 1.

Клієнтська частина програми зображена на рисунку 5.2. Користувач може здійснювани наступні дії:

- відсилати запит на отримання квитка до сервера;
- переглядати доступні ресурси мережі;
- здійснювати доступ до ресурсів корпоративної мережі за допомогою

паролю, квитка та ключа сесії;

– переглядати свої квитки;
– змінювати свій пароль;
– переглянути інформацію про програму.

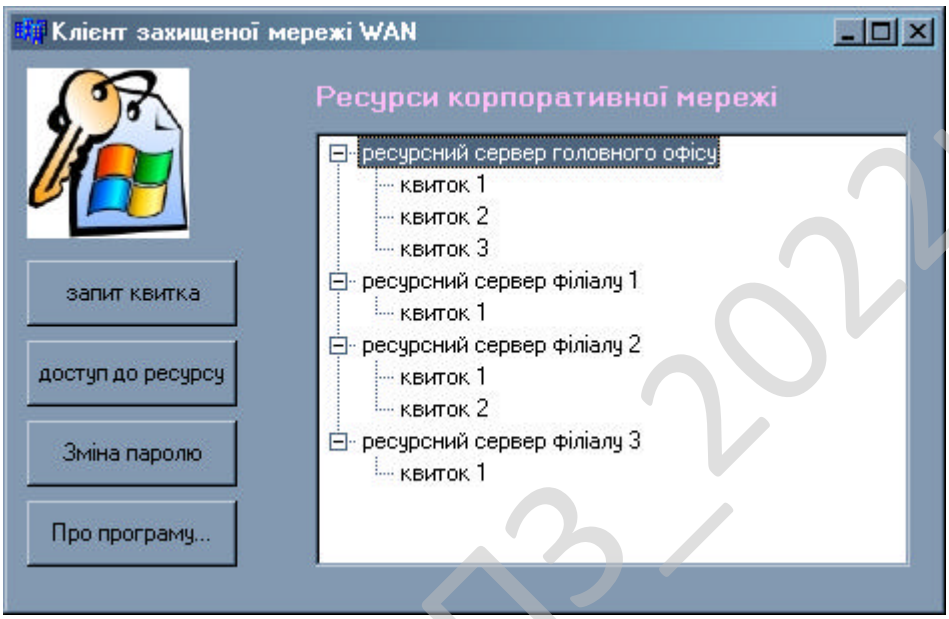

Рисунок 5.2 – Вікно програми-клієнту розробленої системи розподілу ключів корпоративної мережі WAN

Серверна частина програми зображена на рисунку 5.3. Адміністратор має наступні можливості:

– редагування бази даних користувачів;
– зміна параметрів програми;
– зміна паролів;
– редагування бази даних серверів;
– зміна прав доступу;
– перегляд журналу подій;
– перегляд бази даних користувачів, що містить їх логіни, паролі, квитки та ключі;

– перегляд бази даних серверів;

– перегляд інформації про програму.

Вікно "Про програму…" зображене на рисунку 5.4.

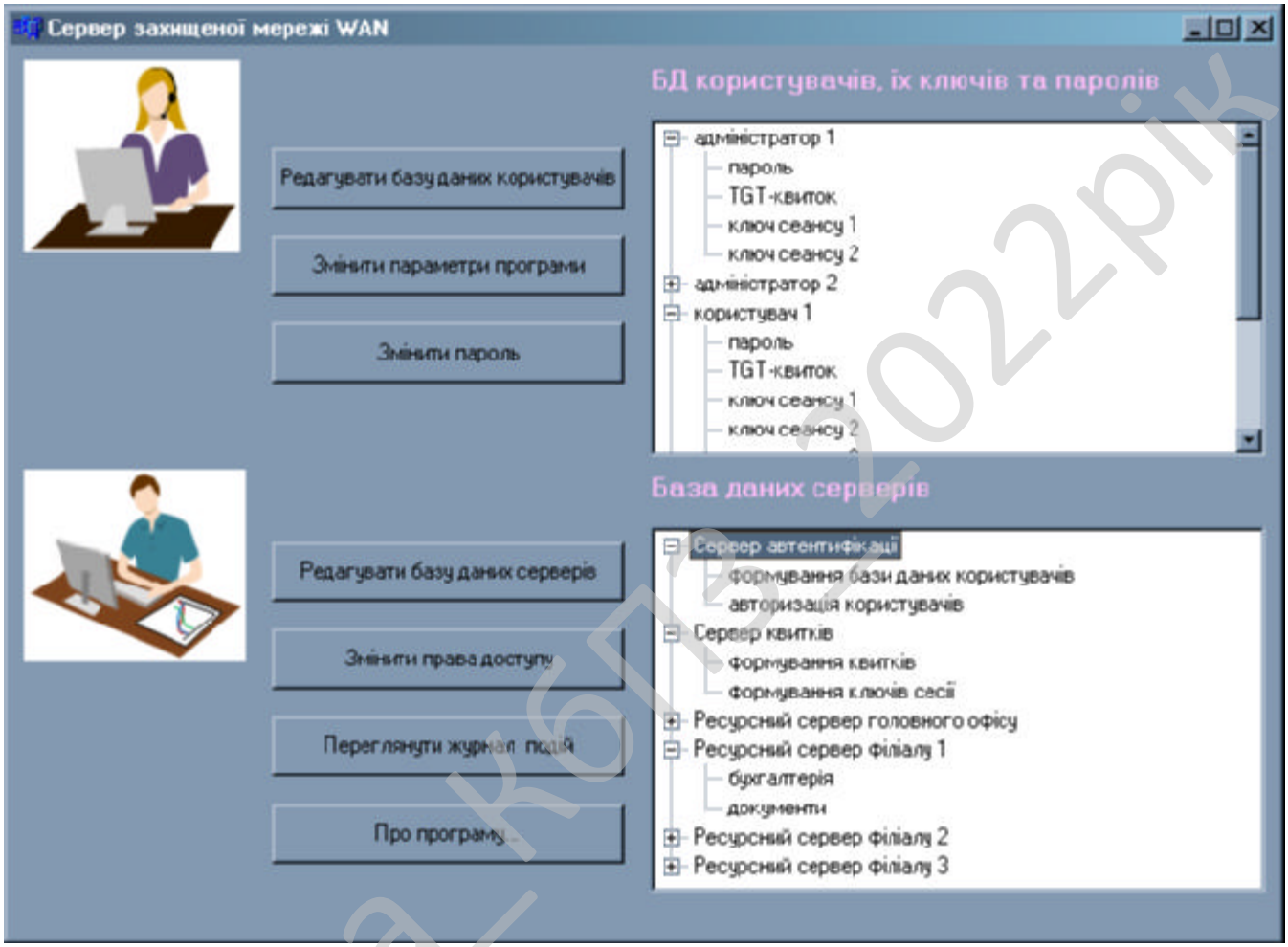

Рисунок 5.3 – Вікно програми-серверу розробленої системи розподілу ключів корпоративної мережі WAN

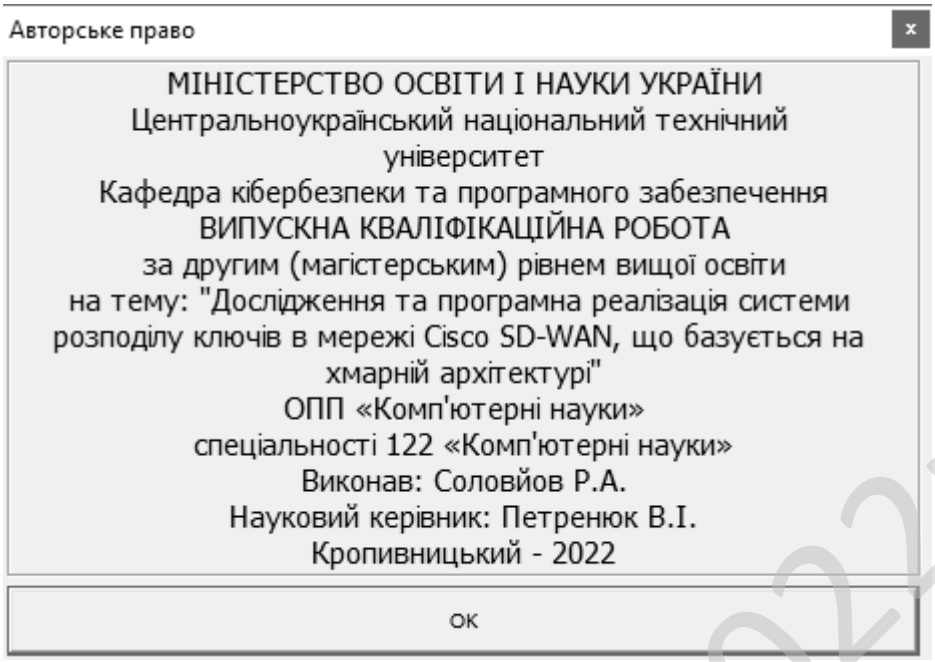

Рисунок 5.4 – Вікно авторського права

Кроки входу користувача в систему:

1. Користувач вводить ім'я й пароль на клієнтській машині.

2. Клієнтська машина виконує над паролем однобічну функцію (гешування), і результат стає секретним ключем клієнта/користувача.

Кроки автентифікації клієнта:

1. Клієнт посилає простим текстом повідомлення серверу AS, запитуючи сервіси від імені користувача. Наприклад так: «Користувач АБВ хоче запросити сервіси». Зверніть увагу, що ні секретний ключ, ні пароль не посилають на AS.

2. AS перевіряє, є чи такий клієнт у базі. Якщо є, то назад AS відправляє наступні два повідомлення:

– Повідомлення A: *Сесійний Ключ Client/TGS* зашифрований секретним ключем клієнта/користувача.

– Повідомлення B: TGT (який включає ID клієнта, мережеву адресу клієнта, період дії квитка, і *Сесійний Ключ Client/TGS*) зашифрований секретним ключем TGS.

3. Як тільки клієнт одержує повідомлення A і B, він розшифровує

повідомлення A, щоб одержати *Сесійний Ключ Client/TGS*. Цей сесійний ключ використовується для подальшого обміну із сервером TGS. (Важливо: Клієнт не може розшифрувати повідомлення B, тому що воно зашифровано секретним ключем TGS.) У цей момент у користувача досить даних, щоб авторизуватися на TGS.

Кроки авторизації клієнта для одержання сервісу:

1. При запиті сервісів клієнт відправляє наступні два повідомлення на TGS:

– Повідомлення C: Містить TGT, отриманий у повідомленні B і ID необхідного сервісу.

– Повідомлення D: Автентифікатор (що складається з ID клієнта й часової мітки), зашифрований на *Сесійному Ключі Client/TGS*.

2. Після одержання повідомлень C і D, TGS витягає повідомлення B з повідомлення C і розшифровує його використовуючи секретний ключ TGS. Це дає йому *Сесійний Ключ Client/TGS*. Використовуючи його TGS розшифровує повідомлення D і посилає наступні два повідомлення клієнтові:

– Повідомлення E: Client-to-*server ticket* (який містить ID клієнта, мережеву адресу клієнта, час дії квитка й *Сесійний Ключ Client/server*) зашифрований секретним ключем сервісу.

– Повідомлення F: *Сесійний ключ Client/server*, зашифрований на *Сесійному Ключі Client/TGS*.

Кроки клієнта при запиті сервісу:

1. При одержанні повідомлень E і F від TGS, у клієнта досить інформації для авторизації на SS. Клієнт з'єднується з SS і посилає наступні два повідомлення:

– Повідомлення E з попереднього кроку (client-to-*server ticket*, зашифрований секретним ключем сервісу).

– Повідомлення G: новий автентифікатор, що включає ID клієнта, часову мітку і зашифрований на client/server session key.

2. SS розшифровує квиток використовуючи свій секретний ключ для одержання *Сесійного Ключа Client/Server*. Використовуючи сесійний ключ, SS розшифровує автентифікатор і посилає клієнтові наступне повідомлення для підтвердження готовності обслужити клієнта й показати, що сервер дійсно є тим, за кого себе видає:

– Повідомлення H: Часова мітка, зазначена клієнтом + 1, зашифрована на *Сесійному Ключі Client/Server*.

3. Клієнт розшифровує підтвердження, використовуючи *Сесійний Ключ Client/Server* і перевіряє, чи дійсно часовий штамп коректно оновлений. Якщо це так, то клієнт може довіряти серверу й може почати посилати запити на сервер.

4. Сервер надає клієнтові необхідний сервіс.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Метою розробки є дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Об'єктом дослідження є процес розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Предметом дослідження є методи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Методи дослідження базуються на методах захисту інформації та хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– Розроблено вітчизняний продукт розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі була досліджена та розроблена програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	20
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	20000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: W_i – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 90 = 151 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	151	Ф 7.1-7.4
Впровадження	13	Д13
Всього	192	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{192 \cdot 1}{60 - 5} = 3,5 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4,17
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	34,83

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{35 \cdot 2}{1{,}2} = 58 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 58/(48 \cdot 8) = 0{,}15 \text{ ставка.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	20000	40000
Продакт-менеджер	0,25	16000	8000
Інженер-програміст	3,5	20000	140000
Інженер-електронщик	0,15	17000	5100
Інженер-системотехнік	0,25	17000	8500
Адміністратор мережі	0,5	17000	17000
Системний програміст	0,25	16000	8000
Дизайнер WEB	0,25	16464	8232
Інженер-верстальник	0,25	16000	8000
Бухгалтер-економіст	0,5	16000	16000
Всього за період розробки	$R_{сп} = 6,9$	-	$Ф_{роб} = 258832$

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{258832}{6,9 \cdot 48} = 781,5 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 10.11.22 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок ПК Fujitsu Esprimo C710 SFF		7347
Процесор	Intel Core i7-3770 (4 (8) ядра по 3.4 - 3. GHz), 8 MB Smart Cache	
Системна плата	Материнська плата Intel Q75 Chipset, 6 USB 2.0, 4x USB 3.0, 2x COM-порт, 2x DV 1x DisplayPort, 2x PS/2, 4x Audio, 1x LAN (RJ-45)	
Відеокарта	nVidia GeForce GT 630	
Жорсткий диск	240 SSD	
Оперативна пам’ять	Kingston 8 GB DDR3	
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	
Корпус	Fujitsu Esprimo C710 SFF, PSU 300W, 12cm black, (front bezel – black+light silver; bod material – 0.6mm), 80mm fan (rear) 2xUSB3.0/ 2xUSB2.0/AUDIO/MIC, Air Duc Tool-less chassis design,Thermall Advantaged Chassis	
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат (МФУ)	1	5965	596,5	6561,5
Всього	–	–	–	199177

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Нематеріальні активи			
4. Нематеріальні активи	20000	10	2000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	5000
Разом	$К_р$ = 1807208		$А_р$ = 176988,5

Примітка: вартість автомобіля Daewoo Lanos 2011 взята по даним сайту https://auto.ria.com/, джерело https://auto.ria.com/uk/auto_daewoo_lanos_33590508.html, складає 143000 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 781{,}5 \cdot 192/20 = 7503 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 7503 \cdot 10 \cdot 0{,}01 = 750{,}3 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(7503+750{,}3) = 1816 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 5):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 22,4 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 22{,}4 \cdot 5 = 112 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 112 + 1702)/20 = 96 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 20 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 176989 \cdot 2/(20 \cdot 12) = 1475 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 7305+730,5+1816+1125+96+1125+1475 = 13672,5 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	7305
2. Додаткова зарплата виконавців	$З_д$	730,5
3. Відрахування на соціальні потреби	$C_{оц}$	1816
4. Загальногосподарські витрати	$Г_{осп}$	1125
5. Витрати на матеріали	$З_М$	96
6. Освоєння нових операційних систем, мов програмування	$О_п$	1125
7. Амортизація основних фондів	$A_м$	1475
8. Повна собівартість програмного забезпечення	$C_п$	13672,5
9. Плановий прибуток	$П_р$	6836,25
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	20508,75
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4101,75
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	24610,5

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \qquad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 13672{,}5 = 6836{,}25 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	24611
Всього капітальних витрат	–	24611

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	45091	22546
2. Витрати на електроенергію	$З_{ел}$	567	284
3. Витрати на амортизацію	$З_{ам}$	0	6153
Всього витрат за рік	$І$	45658	28983

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування сервера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилось з 600 годин на рік до 300 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 600 \cdot 56 \cdot 1{,}1 \cdot 1{,}22 = 45091 \text{ грн},$$

до:

$$З_{р\,нов} = 300 \cdot 56 \cdot 1{,}1 \cdot 1{,}22 = 22546 \text{ грн.}$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\,баз} = 0{,}45 \cdot 600 \cdot 2{,}1 = 567 \text{ грн.}$$

$$З_{ел\,нов} = 0{,}45 \cdot 300 \cdot 2{,}1 = 284 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	24611	–	6152,75
Всього відрахувань	-	–	24611	–	6152,75

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (20508,75 - 13672,5) \cdot 20 - (0,05 \cdot 1408000 + 0,5 \cdot 199177 +$$
$$+0,2 \cdot 143000 + 0,25 \cdot 37031 + 0,1 \cdot 20000) \cdot 2/12 = 101751 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1807208}{(20508,75 - 13672,5) \cdot 20 \cdot 12 / 2} = 2,2 \; років.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	13672,5
3. Ціна розробленої програми	Грн.	20508,75
4. Плановий прибуток від реалізації розробленої програми	Грн.	6836,25
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1807208
7. Загальний прибуток від реалізації програмної продукції	Грн.	136725
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	101751
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,2
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	24611
11.Величина економічного ефекту у користувача програмної продукції	Грн.	10522
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	1,5

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (45658\text{-}28983)\text{-}0{,}25 \cdot 24611 = 10522 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{24611}{45658 - 28983} = 1{,}5 \ роки .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час

роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Аналіз умов праці на робочому місці програміста

Робота програміста пов'язана з постійною роботою на ЕОМ, яка відбувається у кімнаті розмірами 5м×7,2м×2,8м. Одна з її більших стін має шість двостулкових вікон, розмірами 2,2м×1,8м, які виходять на північний схід. Вікна розташовані рівномірно по всій довжині стіни. Підлога в кімнаті покрита лінолеумом, всі стіни пофарбовані світло оранжевого кольору до висоти 2,8м, а далі підвісна стеля. Уздовж стін розташовані комп'ютерні столи. На них розташовуються 2 персональні комп'ютери й інша оргтехніка (сканер принтери, телефони й ксерокс). Столи мають пластикове покриття. Габарити їхньої робочої поверхні 1250 мм×850 мм. Висота столів 750 мм. Висота стільців від рівня підлоги становить 430 мм.

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007 – 98, вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007 – 98 рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А) при

роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765(740)	655(600)	450(440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24°С
	Відносна вологість	40 – 60%
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23 – 25°С
	Відносна вологість	40 ... 60%
	Швидкість руху повітря	0,1 ... 0,2 м / с

8.3 Розробка заходів з умов поліпшення охорони праці

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ.

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і т.д. Подібні крісла,

відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють понизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.4 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5 м, довжина – 7,2 м, висота – 2,8 м.

У зазначеному приміщенні працює 5 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=5 х 7,2 = 36 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами

відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, S = 36 м^2;

h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 5 м;

B – довжина приміщення, B = 7,2 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=1,4.$$

Знаючи індекс приміщення, за знаходимо n = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначимо світловий потік: *F=77478 Лм.*

Для розрахунку будемо використовувати світлодіодні панелі *LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л$ = *3990 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекса приміщення:

$$N= 77478/ 3990=19,4 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 20 шт.

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з умов поліпшення охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– Досліджена система розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

– На основі отриманих результатів досліджень створена програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм DSA.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 10522 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,5 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Соловйов Р.А. Дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. National Institute of Standards and Technology U.S. Department of Commerce "Secure Domain Name System (DNS) Deployment Guide", Special Publication 800-81 (Draft), 2005г, 91с.

3. National Institute of Standards and Technology U.S. Department of Commerce "Intrusion Detection Systems", Special Publication 800-31, 2004г, 51с.

4. National Institute of Standards and Technology U.S. Department of Commerce "Guidelines on Firewalls and Firewall Policy", Special Publication 800-41, 2002г, 74с.

5. National Institute of Standards and Technology U.S. Department of Commerce "Guidelines on Securing Public Web Servers", Special Publication 800-44, 2002г, 187с.

6. RFC 793 "Transmission Control Protocol", 1981г, 85с.

7. RFC 1035 "DOMAIN NAMES – implementation and specification", 1987г., 55с.

8. RFC 2069 "An Extension to HTTP : Digest Access Authentication", 1997г., 18с.

9. RFC 2616 "Hypertext Transfer Protocol -– HTTP/1.1", 1999г., 176с.

10. RFC 2617 "HTTP Authentication: Basic and Digest Access Authentication", 1999г., 34с.

11. RFC 2845 "Secret Key Transaction Authentication for DNS (TSIG)", 2000г., 15с.

12. RFC 2930 "Secret Key Establishment for DNS (TKEY RR)", 2000г., 16с.

13. RFC 2931 "DNS Request and Transaction Signatures (SIG(0)s)", 2000г., 10с.

14. RFC 2964 "Use of HTTP State Management", 2000г., 8с.

15. RFC 2965 "HTTP State Management Mechanism", 2000г., 26с.

16. RFC 3007 "Secure Domain Name System (DNS) Dynamic Update", 2000г., 9с.

17. RFC 3833 "Threat Analysis of the Domain Name System (DNS)", 2004г., 16с.

18. RFC 4033 "DNS Security Introduction and Requirements", 2005г., 21с.

19. RFC 4034 "Resource Records for the DNS Security Extensions", 2005г., 29с.

20. RFC 4035 "Protocol Modifications for the DNS Security Extensions", 2005г., 53с.

21. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

22. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

23. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

24. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence

dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

25. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

26. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

27. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

28. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

29. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

30. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings.* Volume 2740, 2020, Pages 102-114. **(Scopus)**.

31. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

32. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

33. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

34. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

35. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**, «**Representation of Cascade Codes in the Frequency Domain**».** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

36. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus)**.

37. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

38. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

39. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

40. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

41. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. **(Scopus)**.

42. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

43. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019. **(Scopus)**.

44. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629. **(Scopus)**.

45. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

46. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

47. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in

Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

48. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

49. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

50. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

51. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

52. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

53. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022. (**Фахове видання. Категорія «Б»**)

54. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

55. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

56. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

57. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

58. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

59. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв’язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

60. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник “Радіотехніка”* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

61. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

62. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

63. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

64. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь – 2020. – № 3. – С. 50-61.

65. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

66. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

67. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

68. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

69. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

70. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

71. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

72. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

73. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

74. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

75. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

76. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

77. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

78. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

79. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

80. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

81. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

82. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Builder C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута розробка заходів з умов поліпшення охорони праці.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 126 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація системи розподілу ключів в мережі Cisco SD-WAN, що базується на хмарній архітектурі

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 38

Літера: РП

Кропивницький – 2022 року

Основна програма

Файл Project1.cpp основної програми

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//Підключення форм користувача-----------------------------------------------
-------------------------
USEFORM("client.cpp", Form1);
USEFORM("about.cpp", Form2);
USEFORM("avt.cpp", Form3);
USEFORM("server.cpp", Form4);

//Створення форм за допомогою WINAPI ----------------------------------------
--------------------------------

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm3), &Form3);
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->CreateForm(__classid(TForm2), &Form2);
                Application->CreateForm(__classid(TForm4), &Form4);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Файл avt.cpp – авторизація користувача

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "avt.h"
#include "client.h"
#include "server.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button1Click(TObject *Sender)
{

if((Edit1->Text=="1")&(Edit2->Text=="1")) {
Form1->Show();
Form3->Visible=false;
}
if((Edit1->Text=="2")&(Edit2->Text=="2")) {
Form4->Show();
Form3->Visible=false;
}
else   { Label1->Caption="Помилка авторизації";
Label1->Font->Color=clRed;
}
}
//---------------------------------------------------------------------------
```

Файл avt.h – бібліотека для файлу avt.cpp

```
//---------------------------------------------------------------------------

#ifndef avtH
#define avtH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TEdit *Edit1;
        TEdit *Edit2;
        TButton *Button1;
        TImage *Image1;
        TImage *Image2;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

Файл client.cpp програми-клієнту

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "client.h"
#include "avt.h"
#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
//відкриття вікна "про програму..."
Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
// запит квитка клієнтом

int KFW_request_tickets_for_client(char * user)
{
    krb5_context        ctx = 0;
    krb5_error_code         code;
    krb5_principal      princ = 0;
    krb5_ccache             cc  = 0;

    if (!pkrb5_init_context)
        return 0;

    code = pkrb5_init_context(&ctx);
    if (code) return 1;

    code = pkrb5_parse_name(ctx, user, &princ);
    if (code) goto loop_cleanup;

    code = KFW_get_ccache(ctx, princ, &cc);
    if (code) goto loop_cleanup;

    code = pkrb5_cc_destroy(ctx, cc);
    if (!code) cc = 0;

  loop_cleanup:
    if ( cc ) {
        pkrb5_cc_close(ctx, cc);
        cc = 0;
    }
    if ( princ ) {
        pkrb5_free_principal(ctx, princ);
        princ = 0;
    }

    pkrb5_free_context(ctx);
    return 0;
}
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
//доступ до ресурсу клієнтом

//Опис змінних
      char *ignore_str = "--ignore=";
      int ignore_len;
      char *cp, tmp[80];
      char *win_flag;
      char wflags[1024];

#ifdef _WIN32
      win_flag = win32_flag;
#else
      win_flag = "UNIX##";
#endif

      wflags[0] = 0;

//Створення списку ресурсів
      ignore_len = strlen(ignore_str);
      argc--; argv++;
      while (*argv && *argv[0] == '-') {
            wflags[sizeof(wflags) - 1] = '\0';
            if (strlen (wflags) + 1 + strlen (*argv) > sizeof (wflags) - 1) {
                  fprintf (stderr,
                         "wconfig: Дуже багато змінних (internal limit %d)",
                         sizeof (wflags));
                  exit (1);
            }
            if (wflags[0])
                  strcat(wflags, " ");
            strcat(wflags, *argv);

            if (!strcmp(*argv, "--mit")) {
                  mit_specific = 1;
                  argc--; argv++;
                  continue;
            }
            if (!strcmp(*argv, "--win16")) {
                  win_flag = win16_flag;
                  argc--; argv++;
                  continue;
            }
            if (!strcmp(*argv, "--win32")) {
                  win_flag = win32_flag;
                  argc--; argv++;
                  continue;
            }
            if (!strncmp(*argv, "--enable-", 9)) {
                  sprintf(tmp, "%s##", (*argv)+ignore_len);
                  for (cp = tmp; *cp; cp++) {
                        if (islower(*cp))
                              *cp = toupper(*cp);
                  }
                  cp = malloc(strlen(tmp)+1);
                  if (!cp) {
                        fprintf(stderr,
                              "wconfig: malloc failed!\n");
                        exit(1);
                  }
                  strcpy(cp, tmp);
                  add_ignore_list(cp);
                  argc--; argv++;
                  continue;
            }
            if (!strncmp(*argv, ignore_str, ignore_len)) {
```

```
                add_ignore_list((*argv)+ignore_len);
                argc--; argv++;
                continue;
            }
            fprintf(stderr, "Помилка в опціях: %s\n", *argv);
            exit(1);
      }
// Створення списку ресурсів, які заборонені
      if (win_flag)
            add_ignore_list(win_flag);

      if (mit_specific)
            add_ignore_list("MIT##");

      if (wflags[0] && (argc > 0))
            printf("WCONFIG_FLAGS=%s\n", wflags);

      if (argc > 0)
            copy_file (*argv, "win-pre.in");

      copy_file("", "-");

      if (argc > 0)
            copy_file (*argv, "win-post.in");

      return 0;
}

char *ignore_list[64] = {
      "DOS##",
      "DOS",
      };

/*
 * Додавання нових значень до листа ігнорування
 */
void add_ignore_list(char *str)
{
      char **cpp;

      for (cpp = ignore_list; *cpp; cpp++)
            ;
      *cpp = str;
}


/*
 *
 * копіювання файлу доступу
 *
 * копіювання файлу 'path\fname' to stdout.
 *
 */
static int
copy_file (char *path, char *fname)
{
    FILE *fin;
    char buf[1024];
    char **cpp, *ptr;
    int len;

    if (strcmp(fname, "-") == 0) {
          fin = stdin;
    } else {
#ifdef _WIN32
          sprintf(buf, "%s\\%s", path, fname);
#else
          sprintf(buf, "%s/%s", path, fname);
#endif
```

```
          fin = fopen (buf, "r");                    /* File to read */
          if (fin == NULL) {
                fprintf(stderr, "wconfig: файл доступу не відкрито %s\n", buf);
                return 1;
          }
    }


    while (fgets (buf, sizeof(buf), fin) != NULL) { /* копіювання файлу доступу
після завершення роботи */
          if (buf[0] == '@') {
                fputs("\n", stdout);
                continue;
          }
          if (buf[0] != '#' || buf[1] != '#') {
                fputs(buf, stdout);
                continue;
          }
          ptr = buf;
          for (cpp = ignore_list; *cpp; cpp++) {
                len = strlen(*cpp);
                if (memcmp (*cpp, buf+2, len) == 0) {
                      ptr += 2+len;
                      break;
                }
          }
          fputs(ptr, stdout);
    }

    fclose (fin);

    return 0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
//зміна пароля клієнтом
code = pkrb5_change_password_client(ctx,
                                    &my_creds,
                                    me,
                                    password, // пароль
                                    NULL,     //
                                    hParent, //
                                    0, // час початку
                                    0, // ім'я сервісу
                                    &options);
    if (code)
      goto cleanup;

    code = pkrb5_cc_initialize(ctx, cc, me);
    if (code)
      goto cleanup;

    code = pkrb5_cc_store_cred(ctx, cc, &my_creds);
    if (code)
      goto cleanup;
// очищення паролю клієнта
 cleanup:
    if ( addrs ) {
        for ( i=0;i<addr_count;i++ ) {
            if ( addrs[i] ) {
                if ( addrs[i]->contents )
                    free(addrs[i]->contents);
                free(addrs[i]);
            }
        }
    }
//запис нового паролю який генерується системою
    if (my_creds.client == me)
```

```
      my_creds.client = 0;
    pkrb5_free_cred_contents(ctx, &my_creds);
    if (name)
        pkrb5_free_unparsed_name(ctx, name);
    if (me)
        pkrb5_free_principal(ctx, me);
    if (cc && (cc != alt_cc))
        pkrb5_cc_close(ctx, cc);
    if (ctx && (ctx != alt_ctx))
        pkrb5_free_context(ctx);
    return(code);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
Form3->Close();
}
```

Файл client.h – бібліотека для файлу client.cpp

```
//---------------------------------------------------------------------------

#ifndef clientH
#define clientH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TButton *Button2;
        TButton *Button3;
        TTreeView *TreeView1;
        TButton *Button1;
        TButton *Button4;
        TImage *Image1;
        TLabel *Label1;
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
private: // User declarations
public:     // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Файл server.cpp програми-серверу

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "server.h"
#include "avt.h"
#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::FormClose(TObject *Sender, TCloseAction &Action)
{
 Form3->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button1Click(TObject *Sender)
{
//редагування бази даних користувачів
//опис зміних
int argc;
    char *argv[];
{
    extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_newprinc;

    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    extern krb5_kt_ops krb5_ktf_writable_ops;
    int num_to_create;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth;

    krb5_init_context(&test_context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_create = 0;
    depth = 1;

    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
          depth = atoi(optarg);
          break;
        case 'P':
          mkey_password = optarg;
          break;
      case 'p':                         /* посилання на ім’я користувача, що
створюється */
```

```
        strncpy(principal_string, optarg, sizeof(principal_string) - 1);
        principal_string[sizeof(principal_string) - 1] = '\0';
        suffix = principal_string + strlen(principal_string);
        break;
    case 'n':                          /* дуже багато користувачів */
        num_to_create = atoi(optarg);
        break;
    case 'd':               /* установка імені бази даних користувачів */
        dbname = optarg;
        break;
    case 'r':
        cur_realm = optarg;
        break;
    case 'k':
        master_keyblock.enctype = atoi(optarg);
        enctypedone++;
        break;
    case 'M':               /* майстер ключ бази даних користувачів */
        mkey_name = optarg;
        break;
    case 'm':
        manual_mkey = TRUE;
        break;
    case '?':
    default:
        usage(progname, 1);
          }
}

if (!(num_to_create && suffix)) usage(progname, 1);


if ((retval = krb5_kt_register(test_context, &krb5_ktf_writable_ops))) {
    if (retval != KRB5_KT_TYPE_EXISTS) {
    com_err(progname, retval,
        "Реєстрація ключових таблиць бази даних користувачів ");
    exit(1);
  }
}

if (!enctypedone)
  master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;

if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
  com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
        "встановлення типів %d", master_keyblock.enctype);
  exit(1);
}

if (!dbname)
  dbname = DEFAULT_KDB_FILE;

if (!cur_realm) {
  if ((retval = krb5_get_default_realm(test_context, &cur_realm))) {
      com_err(progname, retval, "пошук імені користувача у базі");
      exit(1);
  }
}
if ((retval = set_dbname_help(progname, dbname)))
  exit(retval);

for (n = 1; n <= num_to_create; n++) {
  /* Побудова нового імені користувача */
  /* для початку генеруємо нові імена*/
  (void) sprintf(suffix, "%d", n);
  (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
  tmp[sizeof(tmp) - 1] = '\0';
  str_newprinc = tmp;
  add_princ(test_context, str_newprinc);
```

```
      for (i = 2; i <= depth; i++) {
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_newprinc = tmp;
      add_princ(test_context, str_newprinc);
      }
    }

    retval = krb5_db_fini(test_context);
    memset((char *)master_keyblock.contents, 0,
         (size_t) master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "Кінець наповнення бази даних користувачів");
      exit(1);
    }
    if (master_princ_set) {
      krb5_free_principal(test_context, master_princ);
    }
    krb5_free_context(test_context);
    exit(0);
}

void
add_princ(context, str_newprinc)
    krb5_context    context;
    char          * str_newprinc;
{
    krb5_error_code       retval;
    krb5_principal        newprinc;
    krb5_db_entry         newentry;
    char            princ_name[4096];

    memset((char *)&newentry, 0, sizeof(newentry));
    sprintf(princ_name, "%s@%s", str_newprinc, cur_realm);
    if ((retval = krb5_parse_name(context, princ_name, &newprinc))) {
      com_err(progname, retval, "аналіз імені '%s'", princ_name);
      return;
    }

    /* добавляємо дані на імя користувача */
    newentry.len = KRB5_KDB_V1_BASE_LENGTH;
    newentry.attributes = mblock.flags;
    newentry.max_life = mblock.max_life;
    newentry.max_renewable_life = mblock.max_rlife;
    newentry.expiration = mblock.expiration;
    newentry.pw_expiration = mblock.expiration;

    /* Додаємо характеристики в базу даних */
    if ((retval = krb5_copy_principal(context, newprinc, &newentry.princ))) {
          com_err(progname, retval, "кодуємо дані при запису до бази даних
користувача'%s'",
              princ_name);
      krb5_free_principal(context, newprinc);
      goto error;
    }

    {
      /* Реалізуємо обробку даних */
      krb5_int32 now;

      retval = krb5_timeofday(context, &now);
      if (retval) {
          com_err(progname, retval, "при виборі даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
      retval = krb5_dbe_update_mod_princ_data(context, &newentry, now,
```

```
                                        master_princ);
      if (retval) {
          com_err(progname, retval, "кодуємо дані при додавані до бази даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
    }

    {   /* додаємо ключ користувача до бази даних */
        krb5_data pwd, salt;
        krb5_keyblock key;

        if ((retval = krb5_principal2salt(context, newprinc, &salt))) {
          com_err(progname, retval, "перетворюємо ключ'%s'",
                princ_name);
          krb5_free_principal(context, newprinc);
          goto error;
        }

      krb5_free_principal(context, newprinc);

      pwd.length = strlen(princ_name);
      pwd.data = princ_name;  /* повинно буди регенеруватися */
      if ((retval = krb5_c_string_to_key(context, master_keyblock.enctype,
                                &pwd, &salt, &key))) {
          com_err(progname,retval,"конвертуємо парол до ключа'%s'",
                princ_name);
          krb5_free_data_contents(context, &salt);
          goto error;
      }
      krb5_free_data_contents(context, &salt);

      if ((retval = krb5_dbe_create_key_data(context, &newentry))) {
          com_err(progname, retval, "Створюємо ключові дані користувача '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }

        if ((retval = krb5_dbekd_encrypt_key_data(context,&master_keyblock,
                                      &key, NULL, 1,
                                      newentry.key_data))) {
          com_err(progname, retval, "кодуємо ключові дані для '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }
        free(key.contents);
    }

    {
      int one = 1;

      if ((retval = krb5_db_put_principal(context, &newentry, &one))) {
          com_err(progname, retval, "Запам’ятовуємо дані");
          goto error;
      }
      if (one != 1) {
         com_err(progname,0,"Запис не знайдено в базі користувачів (невідома
помилка)");
          goto error;
      }
    }

    fprintf(stdout, "Додано %s до бази користувачів\n", princ_name);

error: /* Очищуємо змінні */
#if 0
    krb5_dbe_free_contents(context, &newentry);
```

```
#endif
    krb5_db_free_principal(context, &newentry, 1);
    return;
}

int
set_dbname_help(pname, dbname)
char *pname;
char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створюємо й перевіряємо майстер ключ користувача */

    if ((retval = krb5_db_setup_mkey_name(test_context, mkey_name, cur_realm,
                                0, &master_princ))) {
      com_err(pname, retval, "установка імені майстер ключа користувача");
      return(1);
    }
    master_princ_set = 1;
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(test_context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховуємо майстер ключ");
          return(1);
      }
      if ((retval = krb5_c_string_to_key(test_context,
                                master_keyblock.enctype,
                                &pwd, &scratch,
                                &master_keyblock))) {
          com_err(pname, retval,
                "перетворюємо майстер ключ до паролю користуваача");
          return(1);
      }
      free(scratch.data);
    } else {
      if ((retval = krb5_db_fetch_mkey(test_context, master_princ,
                                master_keyblock.enctype, manual_mkey,
                                FALSE, 0, NULL, &master_keyblock))) {
          com_err(pname, retval, "while reading master key");
          return(1);
      }
    }

    if ((retval = krb5_set_default_realm(test_context, cur_realm))) {
      com_err(pname, retval, "встановлюємо встроєну область");
      return 1;
    }
    /* Pathname is passed to db2 via 'args' parameter.  */
    args[1] = NULL;
    args[0] = malloc(sizeof("dbname=") + strlen(dbname));
    if (args[0] == NULL) {
      com_err(pname, errno, "Встановлюємо параметри бази даних користувача");
      return 1;
    }
    sprintf(args[0], "dbname=%s", dbname);

    if ((retval = krb5_db_open(test_context, args, KRB5_KDB_OPEN_RO))) {
      com_err(pname, retval, "ініціалізуємо базу даних користувача");
      return(1);
    }
    free(args[0]);
```

```
    if ((retval = krb5_db_verify_master_key(test_context, master_princ,
                                  &master_keyblock))){
      com_err(pname, retval, "перевіряєм майстер ключ користувача");
      (void) krb5_db_fini(test_context);
      return(1);
    }
    nentries = 1;
    if ((retval = krb5_db_get_principal(test_context, master_princ,
                               &master_entry, &nentries, &more))) {
      com_err(pname, retval, "Пошук основного входу");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (more) {
      com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
            " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (!nentries) {
      com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    }

    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;

    /* не змінюємо прапорці */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(test_context, &master_entry, nentries);
    return 0;
} }
//---------------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
//зміна параметрів програми
}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button4Click(TObject *Sender)
{
//зміна паролю адміністратора


code = pkrb5_change_password_admin(ctx,
                   &my_creds,
                   me,
                   password, // пароль адміністратора
                   NULL,     //
                   hParent, //
                   0, // час початку
                   0, // ім'я сервісу
                   &options);
    if (code)
      goto cleanup;

    code = pkrb5_cc_initialize(ctx, cc, me);
    if (code)
      goto cleanup;

    code = pkrb5_cc_store_cred(ctx, cc, &my_creds);
    if (code)
      goto cleanup;
// очищення паролю адміністратора
 cleanup:
    if ( addrs ) {
        for ( i=0;i<addr_count;i++ ) {
```

```
            if ( addrs[i] ) {
                if ( addrs[i]->contents )
                    free(addrs[i]->contents);
                free(addrs[i]);
            }
        }
    }
//запис нового паролю адміністратора який генерується системою
    if (my_creds.client == me)
      my_creds.client = 0;
    pkrb5_free_cred_contents(ctx, &my_creds);
    if (name)
        pkrb5_free_unparsed_name(ctx, name);
    if (me)
        pkrb5_free_principal(ctx, me);
    if (cc && (cc != alt_cc))
        pkrb5_cc_close(ctx, cc);
    if (ctx && (ctx != alt_ctx))
        pkrb5_free_context(ctx);
    return(code);
}

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender)
{
//редагування бази даних серверів

//опис зміних
int argc;
    char *argv[];
{
    extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_newprinc;

    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    extern krb5_kt_ops krb5_ktf_writable_ops;
    int num_to_create;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth;

    krb5_init_context(&test_context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_create = 0;
    depth = 1;

    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
          depth = atoi(optarg);
          break;
        case 'P':
          mkey_password = optarg;
          break;
      case 'p':                         /* посилання на ім'я серверу, що
створюється */
          strncpy(principal_string, optarg, sizeof(principal_string) - 1);
          principal_string[sizeof(principal_string) - 1] = '\0';
```

```
        suffix = principal_string + strlen(principal_string);
        break;
    case 'n':                          /* дуже багато серверів */
        num_to_create = atoi(optarg);
        break;
    case 'd':               /* установка імені бази даних серверів */
        dbname = optarg;
        break;
    case 'r':
        cur_realm = optarg;
        break;
    case 'k':
        master_keyblock.enctype = atoi(optarg);
        enctypedone++;
        break;
    case 'M':               /* майстер ключ бази даних серверів */
        mkey_name = optarg;
        break;
    case 'm':
        manual_mkey = TRUE;
        break;
    case '?':
    default:
        usage(progname, 1);
          }
}

if (!(num_to_create && suffix)) usage(progname, 1);


if ((retval = krb5_kt_register(test_context, &krb5_ktf_writable_ops))) {
    if (retval != KRB5_KT_TYPE_EXISTS) {
    com_err(progname, retval,
        "Реєстрація ключових таблиць бази даних серверів ");
    exit(1);
  }
}

if (!enctypedone)
  master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;

if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
  com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
        "встановлення типів %d", master_keyblock.enctype);
  exit(1);
}

if (!dbname)
  dbname = DEFAULT_KDB_FILE;

if (!cur_realm) {
  if ((retval = krb5_get_default_realm(test_context, &cur_realm))) {
    com_err(progname, retval, "пошук імені сервера у базі");
    exit(1);
  }
}
if ((retval = set_dbname_help(progname, dbname)))
  exit(retval);

for (n = 1; n <= num_to_create; n++) {
  /* Побудова нового імені сервера */
  /* для початку генеруємо нові імена*/
  (void) sprintf(suffix, "%d", n);
  (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
  tmp[sizeof(tmp) - 1] = '\0';
  str_newprinc = tmp;
  add_princ(test_context, str_newprinc);

  for (i = 2; i <= depth; i++) {
```

```
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_newprinc = tmp;
      add_princ(test_context, str_newprinc);
      }
    }

    retval = krb5_db_fini(test_context);
    memset((char *)master_keyblock.contents, 0,
         (size_t) master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "Кінець наповнення бази даних серверів");
      exit(1);
    }
    if (master_princ_set) {
      krb5_free_principal(test_context, master_princ);
    }
    krb5_free_context(test_context);
    exit(0);
}

void
add_princ(context, str_newprinc)
    krb5_context    context;
    char          * str_newprinc;
{
    krb5_error_code       retval;
    krb5_principal        newprinc;
    krb5_db_entry         newentry;
    char            princ_name[4096];

    memset((char *)&newentry, 0, sizeof(newentry));
    sprintf(princ_name, "%s@%s", str_newprinc, cur_realm);
    if ((retval = krb5_parse_name(context, princ_name, &newprinc))) {
      com_err(progname, retval, "аналіз імені '%s'", princ_name);
      return;
    }

    /* добавляємо дані на імя сервера */
    newentry.len = KRB5_KDB_V1_BASE_LENGTH;
    newentry.attributes = mblock.flags;
    newentry.max_life = mblock.max_life;
    newentry.max_renewable_life = mblock.max_rlife;
    newentry.expiration = mblock.expiration;
    newentry.pw_expiration = mblock.expiration;

    /* Додаємо характеристики в базу даних */
    if ((retval = krb5_copy_principal(context, newprinc, &newentry.princ))) {
          com_err(progname, retval, "кодуємо дані при запису до бази даних
сервера'%s'",
              princ_name);
      krb5_free_principal(context, newprinc);
      goto error;
    }

    {
      /* Реалізуємо обробку даних */
      krb5_int32 now;

      retval = krb5_timeofday(context, &now);
      if (retval) {
          com_err(progname, retval, "при виборі даних");
          krb5_free_principal(context, newprinc);
          goto error;
      }
      retval = krb5_dbe_update_mod_princ_data(context, &newentry, now,
                                      master_princ);
      if (retval) {
```

```
            com_err(progname, retval, "кодуємо дані при додавані до бази даних");
            krb5_free_principal(context, newprinc);
            goto error;
        }
    }

    {   /* додаємо ключ сервера до бази даних */
        krb5_data pwd, salt;
        krb5_keyblock key;

        if ((retval = krb5_principal2salt(context, newprinc, &salt))) {
          com_err(progname, retval, "перетворюємо ключ'%s'",
                princ_name);
          krb5_free_principal(context, newprinc);
          goto error;
        }

      krb5_free_principal(context, newprinc);

      pwd.length = strlen(princ_name);
      pwd.data = princ_name;  /* повинно буди регенеруватися */
      if ((retval = krb5_c_string_to_key(context, master_keyblock.enctype,
                                &pwd, &salt, &key))) {
          com_err(progname,retval,"конвертуємо парол до ключа'%s'",
                princ_name);
          krb5_free_data_contents(context, &salt);
          goto error;
      }
      krb5_free_data_contents(context, &salt);

      if ((retval = krb5_dbe_create_key_data(context, &newentry))) {
          com_err(progname, retval, "Створюємо ключові дані сервера '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }

        if ((retval = krb5_dbekd_encrypt_key_data(context,&master_keyblock,
                                    &key, NULL, 1,
                                    newentry.key_data))) {
          com_err(progname, retval, "кодуємо ключові дані для '%s'",
                princ_name);
            free(key.contents);
          goto error;
        }
        free(key.contents);
    }

    {
      int one = 1;

      if ((retval = krb5_db_put_principal(context, &newentry, &one))) {
          com_err(progname, retval, "Запам’ятовуємо дані");
          goto error;
      }
      if (one != 1) {
         com_err(progname,0,"Запис не знайдено в базі серверів (невідома
помилка)");
          goto error;
      }
    }

    fprintf(stdout, "Додано %s до бази серверів\n", princ_name);

error: /* Очищуємо змінні */
#if 0
    krb5_dbe_free_contents(context, &newentry);
#endif
    krb5_db_free_principal(context, &newentry, 1);
```

```
    return;
}

int
set_dbname_help(pname, dbname)
char *pname;
char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створюємо й перевіряємо майстер ключ сервера */

    if ((retval = krb5_db_setup_mkey_name(test_context, mkey_name, cur_realm,
                                0, &master_princ))) {
      com_err(pname, retval, "установка імені майстер ключа сервера");
      return(1);
    }
    master_princ_set = 1;
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(test_context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховуємо майстер ключ");
          return(1);
      }
      if ((retval = krb5_c_string_to_key(test_context,
                                master_keyblock.enctype,
                                &pwd, &scratch,
                                &master_keyblock))) {
          com_err(pname, retval,
                "перетворюємо майстер ключ до паролю користуваача");
          return(1);
      }
      free(scratch.data);
    } else {
      if ((retval = krb5_db_fetch_mkey(test_context, master_princ,
                           master_keyblock.enctype, manual_mkey,
                           FALSE, 0, NULL, &master_keyblock))) {
          com_err(pname, retval, "while reading master key");
          return(1);
      }
    }

    if ((retval = krb5_set_default_realm(test_context, cur_realm))) {
      com_err(pname, retval, "встановлюємо встроєну область");
      return 1;
    }
    /* Pathname is passed to db2 via 'args' parameter.  */
    args[1] = NULL;
    args[0] = malloc(sizeof("dbname=") + strlen(dbname));
    if (args[0] == NULL) {
      com_err(pname, errno, "Встановлюємо параметри бази даних сервера");
      return 1;
    }
    sprintf(args[0], "dbname=%s", dbname);

    if ((retval = krb5_db_open(test_context, args, KRB5_KDB_OPEN_RO))) {
      com_err(pname, retval, "ініціалізуємо базу даних сервера");
      return(1);
    }
    /* Done with args */
    free(args[0]);

    if ((retval = krb5_db_verify_master_key(test_context, master_princ,
```

```
                                    &master_keyblock))){
      com_err(pname, retval, "перевіряєм майстер ключ сервера");
      (void) krb5_db_fini(test_context);
      return(1);
    }
    nentries = 1;
    if ((retval = krb5_db_get_principal(test_context, master_princ,
                                &master_entry, &nentries, &more))) {
      com_err(pname, retval, "Пошук основного входу");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (more) {
      com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
            " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    } else if (!nentries) {
      com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
      (void) krb5_db_fini(test_context);
      return(1);
    }

    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;

    /* не змінюємо прапорці */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(test_context, &master_entry, nentries);
    return 0;
}
}

}
//---------------------------------------------------------------------------

void __fastcall TForm4::Button5Click(TObject *Sender)
{
//зміна прав доступу

//опис змінних
extern char *optarg;
    int optchar, i, n;
    char tmp[4096], tmp2[BUFSIZ], *str_princ;

    krb5_context context;
    krb5_error_code retval;
    char *dbname = 0;
    int enctypedone = 0;
    int num_to_check;
    char principal_string[BUFSIZ];
    char *suffix = 0;
    int depth, errors;

    krb5_init_context(&context);

    if (strrchr(argv[0], '/'))
      argv[0] = strrchr(argv[0], '/')+1;

    progname = argv[0];

    memset(principal_string, 0, sizeof(principal_string));
    num_to_check = 0;
    depth = 1;
// вибір параметрів
    while ((optchar = getopt(argc, argv, "D:P:p:n:d:r:R:k:M:e:m")) != -1) {
      switch(optchar) {
      case 'D':
```

```
            depth = atoi(optarg);
            break;
          case 'P':
            mkey_password = optarg;
            break;
        case 'p':                          /* префікс імені для перевірки */
            strncpy(principal_string, optarg, sizeof(principal_string) - 1);
            principal_string[sizeof(principal_string) - 1] = '\0';
            suffix = principal_string + strlen(principal_string);
            break;
         case 'n':                           /* як багато для перевірки */
            num_to_check = atoi(optarg);
            break;
        case 'd':                /* установка імені бази даних */
            dbname = optarg;
            break;
        case 'r':
            cur_realm = optarg;
            break;
        case 'k':
            master_keyblock.enctype = atoi(optarg);
            enctypedone++;
            break;
        case 'M':                /* мастер ключ імені в базі даних */
            mkey_name = optarg;
            break;
        case 'm':
            manual_mkey = TRUE;
            break;
        case '?':
        default:
            usage(progname, 1);
            /*Не досягнено бажаного результату*/
        }
      }

      if (!(num_to_check && suffix)) usage(progname, 1);

      if (!enctypedone)
        master_keyblock.enctype = DEFAULT_KDC_ENCTYPE;
// Перевіка валідності ключа користувача
      if (!krb5_c_valid_enctype(master_keyblock.enctype)) {
        com_err(progname, KRB5_PROG_ETYPE_NOSUPP,
              "while setting up enctype %d", master_keyblock.enctype);
        exit(1);
      }

      krb5_use_enctype(context, &master_encblock, master_keyblock.enctype);
// Перевікаа по базі даних
      if (!dbname)
        dbname = DEFAULT_KDB_FILE;

      if (!cur_realm) {
        if ((retval = krb5_get_default_realm(context, &cur_realm))) {
            com_err(progname, retval, "while retrieving default realm name");
            exit(1);
        }
      }
      if ((retval = set_dbname_help(context, progname, dbname)))
        exit(retval);

      errors = 0;

      fprintf(stdout, "\nПеревірка ");

      for (n = 1; n <= num_to_check; n++) {
        /* Змінюємо параметри */
        /* Не вибираємо будь-які імена, тому, що потрібно генерувати імена, які є
в базі даних та до яких є ключ */
```

```
      (void) sprintf(suffix, "%d", n);
      (void) sprintf(tmp, "%s-DEPTH-1", principal_string);
      str_princ = tmp;
      if (check_princ(context, str_princ)) errors++;

      for (i = 2; i <= depth; i++) {
      (void) sprintf(tmp2, "/%s-DEPTH-%d", principal_string, i);
      tmp2[sizeof(tmp2) - 1] = '\0';
      strncat(tmp, tmp2, sizeof(tmp) - 1 - strlen(tmp));
      str_princ = tmp;
      if (check_princ(context, str_princ)) errors++;
      }
    }

    if (errors)
      fprintf(stdout, "\n%d Помилка.\n", errors);
    else
      fprintf(stdout, "\n Помилки немає.\n");

    krb5_finish_random_key(context, &master_encblock, &master_random);
    krb5_finish_key(context, &master_encblock);

    retval = krb5_db_fini(context);
    memset((char *)master_keyblock.contents, 0, (size_t)
master_keyblock.length);
    if (retval && retval != KRB5_KDB_DBNOTINITED) {
      com_err(progname, retval, "При закритті бази даних");
      exit(1);
    }

    if (str_master_princ) {
      krb5_free_unparsed_name(context, str_master_princ);
    }
    krb5_free_principal(context, master_princ);
    krb5_free_context(context);
    exit(0);
}

int
check_princ(context, str_princ)
    krb5_context context;
    char * str_princ;
{
    krb5_error_code retval;
    krb5_db_entry kdbe;
    krb5_keyblock pwd_key, db_key;
    krb5_data pwd, salt;
    krb5_principal princ;
    krb5_boolean more;
    int nprincs = 1;
    /* char *str_mod_name; */
    char princ_name[4096];

    sprintf(princ_name, "%s@%s", str_princ, cur_realm);

    fprintf(stderr, "\t%s ...\n", princ_name);

    if ((retval = krb5_parse_name(context, princ_name, &princ))) {
      com_err(progname, retval, "while parsing '%s'", princ_name);
      goto out;
    }

    pwd.data = princ_name;  /* Повинно бути в змозі змінюватися */
    pwd.length = strlen(princ_name);

    if ((retval = krb5_principal2salt(context, princ, &salt))) {
      com_err(progname, retval, "При перетворенні принципових параметрів
розподілу ключів'%s'", princ_name);
      krb5_free_principal(context, princ);
```

```
      goto out;
    }

    if ((retval = krb5_string_to_key(context, &master_encblock,
                            &pwd_key, &pwd, &salt))) {
      com_err(progname, retval, "при перетворенні паролю в ключ для '%s'",
            princ_name);
      krb5_free_data_contents(context, &salt);
      krb5_free_principal(context, princ);
      goto out;
    }
    krb5_free_data_contents(context, &salt);

    if ((retval = krb5_db_get_principal(context, princ, &kdbe,
                              &nprincs, &more))) {
      com_err(progname, retval, "Перевіка існування параметрів та прав
доступу");
      krb5_free_principal(context, princ);
      goto out;
    }
    krb5_free_principal(context, princ);

    if (nprincs != 1) {
      com_err(progname, 0, "Знайдено %d входів в базу даних %s.\n", nprincs,
            princ_name);
      goto errout;
    }

    if ((retval = krb5_dbekd_decrypt_key_data(context, &master_keyblock,
                                        kdbe.key_data, &db_key, NULL))) {
      com_err(progname, retval, "Ключ декодується для '%s'", princ_name);
      goto errout;
    }

    if ((pwd_key.enctype != db_key.enctype) ||
      (pwd_key.length != db_key.length)) {
      fprintf (stderr, "\t Ключові типи не співпадають (%d expected, %d from
db)\n",
             pwd_key.enctype, db_key.enctype);
errout:
      krb5_db_free_principal(context, &kdbe, nprincs);
      return(-1);
    }
    else {
      if (memcmp((char *)pwd_key.contents, (char *) db_key.contents,
            (size_t) pwd_key.length)) {
      fprintf(stderr, "\t Ключ не відповідає збереженій величині для %s\n",
            princ_name);
      goto errout;
      }
    }

    free((char *)pwd_key.contents);
    free((char *)db_key.contents);

    if (kdbe.key_data[0].key_data_kvno != 1) {
      fprintf(stderr,"\t kvno не відповідає збереженій величині для %s.\n",
princ_name);
      goto errout;
    }

    if (kdbe.max_life != mblock.max_life) {
      fprintf(stderr, "\tmax не відповідає збереженій величині для %s.\n",
            princ_name);
      goto errout;
    }

    if (kdbe.max_renewable_life != mblock.max_rlife) {
      fprintf(stderr,
```

```
            "\tmax не відповідає збереженій величині для %s.\n",
            princ_name);
      goto errout;
    }

    if (kdbe.expiration != mblock.expiration) {
      fprintf(stderr, "\Час texpiration не відповідає збереженій величині для
%s.\n",
            princ_name);
      goto errout;
    }

/*
    if ((retval = krb5_unparse_name(context, kdbe.mod_name, &str_mod_name)))
      com_err(progname, retval, "для непарсингу імені режиму");
    else {
      if (strcmp(str_mod_name, str_master_princ) != 0) {
      fprintf(stderr, "\tmod не є визначеним іменем  (%s not %s).\n",
            str_mod_name, str_master_princ);
      free(str_mod_name);
      goto errout;
      }
      else free(str_mod_name);
    }
*/

    if (kdbe.attributes != mblock.flags) {
      fprintf(stderr, "\tАтрибути не відповідають збереженій величині для
%s.\n",
            princ_name);
      goto errout;
    }

    out:
    krb5_db_free_principal(context, &kdbe, nprincs);

    return(0);
}

int
set_dbname_help(context, pname, dbname)
    krb5_context context;
    char *pname;
    char *dbname;
{
    krb5_error_code retval;
    int nentries;
    krb5_boolean more;
    krb5_data pwd, scratch;
    char *args[2];

    /* створення нового ключа */

    if ((retval = krb5_db_setup_mkey_name(context, mkey_name, cur_realm, 0,
                                &master_princ))) {
      com_err(pname, retval, "при встановленні імені ключа");
      return(1);
    }
    if (mkey_password) {
      pwd.data = mkey_password;
      pwd.length = strlen(mkey_password);
      retval = krb5_principal2salt(context, master_princ, &scratch);
      if (retval) {
          com_err(pname, retval, "розраховується майстер ключ");
          return(1);
      }
      if ((retval = krb5_string_to_key(context, &master_encblock,
                            &master_keyblock, &pwd, &scratch))) {
          com_err(pname, retval,
```

```
            "перетворюємо майстер ключ в пароль");
        return(1);
    }
    free(scratch.data);
  } else {
    if ((retval = krb5_db_fetch_mkey(context, master_princ,
                             master_keyblock.enctype,
                            manual_mkey, FALSE, (char *) NULL, 0,
                            &master_keyblock))) {
        com_err(pname, retval, "Зчитуємо майсер ключ");
        return(1);
    }
  }

  /* Поточний інтерфейс вимагає, щоб default_realm
     Поле було встановлено в  krb5_context.  */
  if ((retval = krb5_set_default_realm(context, cur_realm))) {
    com_err(pname, retval, "Відповідні установки");
    return 1;
  }
  /* Захищений путь до  db2 з 'args' параметрами.  */
  args[1] = NULL;
  args[0] = malloc(sizeof("dbname=") + strlen(dbname));
  if (args[0] == NULL) {
    com_err(pname, errno, "Установлюємо параметри бази даних");
    return 1;
  }
  sprintf(args[0], "dbname=%s", dbname);

  if ((retval = krb5_db_open(context, args, KRB5_KDB_OPEN_RO))) {
    com_err(pname, retval, "Ініціалізуємо базу даних");
    return(1);
  }
  if ((retval = krb5_db_verify_master_key(context, master_princ,
                                  &master_keyblock))) {
    com_err(pname, retval, "Перевіряємо майстер ключ");
    (void) krb5_db_fini(context);
    return(1);
  }
  nentries = 1;
  if ((retval = krb5_db_get_principal(context, master_princ, &master_entry,
                           &nentries, &more))) {
    com_err(pname, retval, "Шукаємо текст ");
    (void) krb5_db_fini(context);
    return(1);
  } else if (more) {
    com_err(pname, KRB5KDC_ERR_PRINCIPAL_NOT_UNIQUE,
          "Пошук основного входу");
    (void) krb5_db_fini(context);
    return(1);
  } else if (!nentries) {
    com_err(pname, KRB5_KDB_NOENTRY, " Пошук основного входу ");
    (void) krb5_db_fini(context);
    return(1);
  }

  if ((retval = krb5_unparse_name(context, master_princ,
                         &str_master_princ))) {
    com_err(pname, retval, "Помилка пошуку");
    krb5_db_fini(context);
    return(1);
  }

  if ((retval = krb5_process_key(context,
                       &master_encblock, &master_keyblock))) {
    com_err(pname, retval, "Обробка майстер ключа");
    (void) krb5_db_fini(context);
    return(1);
  }
```

```
    if ((retval = krb5_init_random_key(context,
                              &master_encblock, &master_keyblock,
                              &master_random))) {
      com_err(pname, retval, "ініціалізуємо генератор випадкових ключів");
      krb5_finish_key(context, &master_encblock);
      (void) krb5_db_fini(context);
      return(1);
    }
    mblock.max_life = master_entry.max_life;
    mblock.max_rlife = master_entry.max_renewable_life;
    mblock.expiration = master_entry.expiration;
    /* Не треба встановлювати інші флаги */
    mblock.mkvno = master_entry.key_data[0].key_data_kvno;

    krb5_db_free_principal(context, &master_entry, nentries);
    return 0; }
//---------------------------------------------------------------------------

void __fastcall TForm4::Button6Click(TObject *Sender)
{
//перегляд журналу подій
void
kmqint_dump_publisher(FILE * f) {

    int n_free = 0;
    int n_active = 0;
    kmq_message * m;

    EnterCriticalSection(&cs_kmq_msg);

    fprintf(f, "qp0\t*** Події ***\n");
    fprintf(f, "qp1\t Адреса\n");

    m = msg_free;
    while(m) {
        n_free++;

        fprintf(f, "qp2\t0x%p\n", m);

        m = LNEXT(m);
    }

    fprintf(f, "qp3\t Всього подій : %d\n", n_free);

    fprintf(f, "qp4\t*** Активних подій ***\n");
    fprintf(f,
"qp5\tAddress\tType\tSubtype\tuParam\tvParam\tnSent\tnCompleted\tnFailed\twait_o
\trefcount\n");

    m = msg_active;
    while(m) {

        n_active++;

        fprintf(f, "qp6\t0x%p\t%d\t%d\t0x%x\t0x%p\t%d\t%d\t%d\t0x%p\t%d\n",
                m,
                (int) m->type,
                (int) m->subtype,
                (unsigned int) m->uparam,
                m->vparam,
                (int) m->nSent,
                (int) m->nCompleted,
                (int) m->nFailed,
                (void *) m->wait_o,
                (int) m->refcount);

        m = LNEXT(m);
    }
```

```
    fprintf(f, "qp7\t Кількість активних подій = %d\n", n_active);

    fprintf(f, "qp8\t--- End ---\n");

    LeaveCriticalSection(&cs_kmq_msg);

}

#endif

/*! Кратке резюме дій об'єкту події */
kmq_message *
kmqint_get_message(void) {
    kmq_message * m;

    LPOP(&msg_free,&m);
    if(!m) {
        m = PMALLOC(sizeof(kmq_message));
    }
    ZeroMemory((void*)m, sizeof(kmq_message));

    LPUSH(&msg_active, m);

    return m;
}

void
kmqint_put_message(kmq_message *m) {
    int queued;
    /* Ми повинні звільнити місце под подію. У іншому випадку ми повинні чекати
поки подія не буде завершена */
    if(m->refcount == 0) {
        LDELETE(&msg_active, m);
        LeaveCriticalSection(&cs_kmq_msg);
        queued = kmqint_notify_msg_completion(m);
        EnterCriticalSection(&cs_kmq_msg);
        if (!queued) {
            if(m->err_ctx) {
                kherr_release_context(m->err_ctx);
                m->err_ctx = NULL;
            }
            if(m->wait_o) {
                CloseHandle(m->wait_o);
                m->wait_o = NULL;
            }
            LPUSH(&msg_free,m);
        }
    } else if(m->wait_o) {
        SetEvent(m->wait_o);
    }
}

/*! Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_send_message(khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * blob) {
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_message_ex(type, subtype, uparam, blob, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);
```

```
    return rv;
}

/*! \Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_post_message(khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * blob) {
    return kmqint_post_message_ex(type, subtype, uparam, blob, NULL, FALSE);
}

/*! \Отримуємо ::cs_kmq_msg
    */
KHMEXP khm_int32 KHMAPI
kmq_free_call(kmq_call call) {
    kmq_message * m;

    m = call;

    EnterCriticalSection(&cs_kmq_msg);
    m->refcount--;
    if(!m->refcount) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);

    return KHM_ERROR_SUCCESS;
}

/*! \Отримуємо ::cs_kmq_msg, ::cs_kmq_types, ::cs_kmq_msg_ref, kmq_queue::cs
    */
khm_int32
kmqint_post_message_ex(khm_int32 type, khm_int32 subtype, khm_ui_4 uparam,
    void * blob, kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = blob;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    kmqint_msg_publish(m, try_send);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_message_ex(khm_int32 type, khm_int32 subtype,
                    khm_ui_4 uparam, void * blob, kmq_call * call)
{
    return kmqint_post_message_ex(type, subtype, uparam, blob, call, FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_abort_call(kmq_call call)
{
    return KHM_ERROR_NOT_IMPLEMENTED;
}

KHMEXP khm_int32 KHMAPI
kmq_post_sub_msg(khm_handle sub, khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * vparam)
{
    return kmq_post_sub_msg_ex(sub, type, subtype, uparam, vparam, NULL);
}

khm_int32
kmqint_post_sub_msg_ex(khm_handle sub, khm_int32 type, khm_int32 subtype,
                       khm_ui_4 uparam, void * vparam,
                       kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = vparam;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
            /* leave it held */
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
    EnterCriticalSection(&cs_kmq_msg);
    kmqint_post((kmq_msg_subscription *) sub, m, try_send);

    if(m->nCompleted + m->nFailed == m->nSent) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);
```

```
    if (try_send)
        LeaveCriticalSection(&cs_kmq_types);

    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_sub_msg_ex(khm_handle sub, khm_int32 type, khm_int32 subtype,
                    khm_ui_4 uparam, void * vparam, kmq_call * call)
{
    return kmqint_post_sub_msg_ex(sub, type, subtype,
                                  uparam, vparam, call, FALSE);
}

khm_int32
kmqint_post_subs_msg_ex(khm_handle * subs, khm_size   n_subs, khm_int32 type,
                        khm_int32 subtype, khm_ui_4 uparam, void * vparam,
                        kmq_call * call, khm_boolean try_send)
{
    kmq_message * m;
    kherr_context * ctx;
    khm_size i;

    if(n_subs == 0)
        return KHM_ERROR_SUCCESS;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = type;
    m->subtype = subtype;
    m->uparam = uparam;
    m->vparam = vparam;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    ctx = kherr_peek_context();
    if (ctx) {
        if (ctx->flags & KHERR_CF_TRANSITIVE) {
            m->err_ctx = ctx;
            /* leave it held */
        } else {
            kherr_release_context(ctx);
        }
    }

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
    EnterCriticalSection(&cs_kmq_msg);
    for(i=0;i<n_subs;i++) {
        kmqint_post((kmq_msg_subscription *) subs[i], m, try_send);
    }

    if(m->nCompleted + m->nFailed == m->nSent) {
        kmqint_put_message(m);
    }
    LeaveCriticalSection(&cs_kmq_msg);
    if (try_send)
        EnterCriticalSection(&cs_kmq_types);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_post_subs_msg(khm_handle * subs,
                  khm_size   n_subs,
                  khm_int32 type,
                  khm_int32 subtype,
                  khm_ui_4 uparam,
                  void * vparam)
{
    return kmqint_post_subs_msg_ex(subs,
                                   n_subs,
                                   type,
                                   subtype,
                                   uparam,
                                   vparam,
                                   NULL,
                                   FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_post_subs_msg_ex(khm_handle * subs,
                     khm_int32 n_subs,
                     khm_int32 type,
                     khm_int32 subtype,
                     khm_ui_4 uparam,
                     void * vparam,
                     kmq_call * call)
{
    return kmqint_post_subs_msg_ex(subs, n_subs, type, subtype,
                                   uparam, vparam, call, FALSE);
}

KHMEXP khm_int32 KHMAPI
kmq_send_subs_msg(khm_handle *subs,
                  khm_int32 n_subs,
                  khm_int32 type,
                  khm_int32 subtype,
                  khm_ui_4 uparam,
                  void * vparam)
{
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_subs_msg_ex(subs, n_subs, type, subtype,
                                 uparam, vparam, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);

    return rv;
}

KHMEXP khm_int32 KHMAPI
kmq_send_sub_msg(khm_handle sub, khm_int32 type, khm_int32 subtype,
                 khm_ui_4 uparam, void * vparam)
{
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmqint_post_sub_msg_ex(sub, type, subtype, uparam, vparam, &c, TRUE);
    if(KHM_FAILED(rv))
        return rv;
```

```
    rv = kmq_wait(c, INFINITE);
    if(KHM_SUCCEEDED(rv) && c->nFailed > 0)
        rv = KHM_ERROR_PARTIAL;

    kmq_free_call(c);

    return rv;
}

/*! Отримуємо ::cs_kmq_global, ::cs_kmq_msg, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_send_thread_quit_message(kmq_thread_id thread, khm_ui_4 uparam) {
    kmq_call c;
    khm_int32 rv = KHM_ERROR_SUCCESS;

    rv = kmq_post_thread_quit_message(thread, uparam, &c);
    if(KHM_FAILED(rv))
        return rv;

    rv = kmq_wait(c, INFINITE);

    kmq_free_call(c);

    return rv;
}

/*! Отримуємо ::cs_kmq_global, ::cs_kmq_msg, ::cs_kmq_msg_ref, kmq_queue::cs
    */
KHMEXP khm_int32 KHMAPI
kmq_post_thread_quit_message(kmq_thread_id thread,
                             khm_ui_4 uparam, kmq_call * call) {
    kmq_message * m;
    kmq_queue * q;

    EnterCriticalSection(&cs_kmq_global);
    q = queues;
    while(q) {
        if(q->thread == thread)
            break;
        q = LNEXT(q);
    }
    LeaveCriticalSection(&cs_kmq_global);

    if(!q)
        return KHM_ERROR_NOT_FOUND;

    EnterCriticalSection(&cs_kmq_msg);
    m = kmqint_get_message();
    LeaveCriticalSection(&cs_kmq_msg);

    m->type = KMSG_SYSTEM;
    m->subtype = KMSG_SYSTEM_EXIT;
    m->uparam = uparam;
    m->vparam = NULL;

    m->timeSent = GetTickCount();
    m->timeExpire = m->timeSent + kmq_call_dead_timeout;

    if(call) {
        m->wait_o = CreateEvent(NULL,FALSE,FALSE,NULL);
        *call = m;
        m->refcount++;
    } else
        m->wait_o = NULL;

    kmqint_post_queue(q, m);
```

```
    return KHM_ERROR_SUCCESS;
}

KHMEXP khm_int32 KHMAPI
kmq_get_next_response(kmq_call call, void ** resp) {
    return 0;
}

KHMEXP khm_boolean KHMAPI
kmq_has_completed(kmq_call call) {
    khm_boolean completed;

    EnterCriticalSection(&cs_kmq_msg);
    completed = (call->nCompleted + call->nFailed == call->nSent);
    LeaveCriticalSection(&cs_kmq_msg);

    return completed;
}

KHMEXP khm_int32 KHMAPI
kmq_wait(kmq_call call, kmq_timer timeout) {
    kmq_message * m = call;
    DWORD rv;

    if(m && m->wait_o) {
        rv = WaitForSingleObject(m->wait_o, timeout);
        if(rv == WAIT_OBJECT_0)
            return KHM_ERROR_SUCCESS;
        else
            return KHM_ERROR_TIMEOUT;
    } else
        return KHM_ERROR_INVALID_PARAM;
}

/*! \Отримуємо ::cs_kmq_types
    */
KHMEXP khm_int32 KHMAPI
kmq_set_completion_handler(khm_int32 type,
                           kmq_msg_completion_handler handler) {
    return kmqint_msg_type_set_handler(type, handler);
}

//---------------------------------------------------------------------------

void __fastcall TForm4::Button7Click(TObject *Sender)
{
//відкриття вікна "про програму..."
Form2->Show();
}
//---------------------------------------------------------------------------
```

Файл server.h – бібліотека для файлу server.cpp

```
//---------------------------------------------------------------------------

#ifndef serverH
#define serverH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm4 : public TForm
{
__published:     // IDE-managed Components
        TTreeView *TreeView1;
        TLabel *Label1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TTreeView *TreeView2;
        TLabel *Label2;
        TButton *Button5;
        TButton *Button6;
        TImage *Image1;
        TImage *Image3;
        TButton *Button7;
        void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
private:          // User declarations
public:           // User declarations
        __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```

Файл about.cpp – вікно "Про програму..."

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "about.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------------
```

Файл about.h – бібліотека для файлу about.cpp

```
//---------------------------------------------------------------------------

#ifndef aboutH
#define aboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-managed Components
        TImage *Image1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```