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 проектування сервісів автентифікації”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Середа М.Л.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 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 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

Середі Максиму Леонідовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи проектування сервісів автентифікації*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 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 проектування сервісів автентифікації*

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
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Середа М.Л.
(прізвище та ініціали)

АНОТАЦІЯ

Середа М.Л. Дослідження та програмна реалізація системи проектування сервісів автентифікації. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи проектування сервісів автентифікації.

Метою розробки є дослідження та програмна реалізація системи проектування сервісів автентифікації.

Об'єктом дослідження є процес проектування сервісів автентифікації.

Предметом дослідження є методи проектування сервісів автентифікації.

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

Результат роботи – програмна реалізація системи проектування сервісів автентифікації.

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 Delphi 10.4.1.

Ключові слова: комп'ютерна інженерія, автентифікація

ABSTRACT

Sereda M.L. Research and software implementation of the design system of authentication services. 123 Computer engineering. 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 design system of authentication services.

The purpose of the development is research and software implementation of the design system of authentication services.

The object of research is the process of designing authentication services.

The subject of research is methods of designing authentication services.

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

The result of the work is the software implementation of the authentication service design system.

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 Delphi 10.4.1 environment.

Keywords: computer engineering, authentication

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕЦП	–	електронний цифровий підпис
ІТ	–	інформаційні технології
КУМЗ	–	класи уніфікованих математичних завдань
ОМЗ	–	основні математичні завдання
ПЗ	–	програмне забезпечення
СКЗІ	–	система контролю та захисту інформації
СКУД	–	система контролю й управління доступом у приміщення
ОАОР	–	загальний річний ріст
OTP	–	OneTime Password
PKI	–	інфрастуктура відкритих ключів
USB	–	universal serial bus

ВСТУП

Актуальність проблеми. На сучасному рівні розвитку обчислювальної техніки кожний користувач інформаційних систем зіштовхується із процедурами: "ідентифікація" і "автентифікація", неодноразово протягом робочого дня.

Ці процедури виконуються щораз, коли користувач уводить пароль для доступу до комп'ютера, у мережу, до бази даних або при запуску прикладної програми. У результаті їхнього виконання він одержує або доступ до ресурсу, або відмову в доступі.

Строго кажучи, цей процес складається із двох частин – ідентифікації й автентифікації. Ідентифікація – це пред'явлення користувачем якогось унікального, властивого тільки йому ознаки-ідентифікатора. Це може бути пароль, якась біометрична інформація, наприклад відбиток пальця, персональний електронний ключ або смарт-карта й т.і. Автентифікація – це процедура, що перевіряє, чи має користувач із пред'явленим ідентифікатором право на доступ до ресурсу.

Ці процедури нерозривно зв'язані між собою, оскільки спосіб перевірки визначає, яким образом і що користувач повинен пред'явити системі, щоб одержати доступ.

У даній магістерській роботі буде розглянута автентифікація з використанням електронних ключів, більш конкретно з використанням як ключ USB-накопичувача.

Даний метод автентифікації припускає використання електронних ключів для зберігання облікових записів користувачів.

У цьому варіанті використовується звичайний метод автентифікації з використанням паролів, однак паролі зберігаються не в голові користувача, а в пам'яті електронного ключа або смарт-карти, і вводяться не вручну із клавіатури, а зчитуються з електронного ключа при його приєднанні до комп'ютера.

Для реалізації такого методу автентифікації на клієнтські комп'ютери встановлюється спеціальний драйвер, що реагує на приєднання електронного ключа, запитує PIN-код, зчитує з пам'яті ключа профіль для входу в комп'ютер, тобто ім'я користувача, пароль, ім'я домена або дерева eDirectory і здійснює вхід у комп'ютер з використанням прочитаних даних.

Процес видачі ключа користувачеві досить простий: адміністратор повинен створити профіль доступу й записати його до пам’яті ключа. Для цього адміністратор вибирає облікові записи користувача в службах каталогу Windows і NetWare, для них генеруються нові паролі, записуються в ключ і міняються у відповідних службах каталогу.

Пароль при цьому генерується автоматично й відразу записується до пам’яті ключа, користувач його навіть не знає. Це дає можливість генерації «сильних» паролів, що представляють собою довільну комбінацію букв, цифр і спецсимволів.

Крім цього, існує можливість автоматичної зміни паролів з будь-якою періодичністю, наприклад при кожному вході в систему. Цей процес також відбувається без участі користувача.

В одному профілі може зберігатися інформація про один обліковий запис в Windows і однієї – в NetWare. У ключі може бути кілька профілів, у цьому випадку драйвер при приєднанні ключа відобразить список профілів і запропонує користувачеві вибрати один з них. Це зручно, якщо користувач може здійснювати вхід у мережу під різними обліковими записами з різними правами.

Такий метод автентифікації не підтримується стандартними засобами операційних систем. Проте, деякі виробники електронних ключів поставляють у комплекті SDK підсистему автентифікації, що реалізує описаний функціонал, однак, як правило, такі реалізації однобокі й носять більшою мірою ознайомлювальний характер.

Єдиний варіант – скористатися спеціально розробленою в результаті виконання магістерської роботи системою автентифікації з використанням

електронних ключів.

Основна перевага розробленої системи системи полягає в мінімальній перебудові IT-інфраструктури організації, мінімальні витрати на адміністрування, одночасної автентифікації в службах каталогу Windows і NetWare, підвищеної безпека мережі за рахунок використання "сильних" паролів і зберігання їх у захищеній пам'яті електронних ключів.

Крім цього, істотно знижується "людський фактор", оскільки користувач просто не знає пароля, тому не може його нікому передати або записати на папірці.

Стрімкий розвиток засобів захисту інформації й методів їхнього аналізу в ряді випадків приводить до зміни погляду на безпеку існуючої апаратури захисту інформації, перегляду моделі порушника й підвищення нормативних вимог. Найбільшою мірою зазначені зміни характерні для підсистем автентифікації, у яких виникає необхідність забезпечення інформаційної безпеки в умовах взаємної недовіри або змови учасників протоколу, що дозволяє говорити про критичні підсистеми автентифікації. Найважливішим етапом створення таких підсистем є проектування алгоритмів автентифікації.

Безпека підсистем автентифікації не може бути перевірена експериментально в ході випробувань на функціонування. Крім того, через достаток криптографічних алгоритмів і різноманіття завдань автентифікації ці системи найчастіше проектуються «з нуля», що збільшує трудомісткість розробки. Алгоритмізація процесу проектування й обґрунтування прийнятих рішень дозволять уникнути типових помилок, а також порівнювати різні варіанти побудови алгоритмів автентифікації й вибирати найкращий з них.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи проектування сервісів автентифікації.

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

– Огляд існуючих систем проектування сервісів автентифікації.

– Дослідження системи проектування сервісів автентифікації.

– Програмна реалізація системи проектування сервісів автентифікації.

Об'єктом дослідження є процес проектування сервісів автентифікації.

Предметом дослідження є методи проектування сервісів автентифікації.

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

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

– Удосконалено метод проектування сервісів автентифікації.

– Розроблено вітчизняний продукт проектування сервісів автентифікації, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі проектування сервісів автентифікації.

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 проектування сервісів автентифікації, 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 призначена для реалізації проектування сервісів автентифікації. Розглянемо поняття автентифікації.

Автентифікація – перевірка приналежності суб'єктові доступу пред'явленого їм ідентифікатора; підтвердження дійсності.

Автентифікацію не слід плутати з ідентифікацією. Автентифікатор – це пакет, що доводить, що клієнт дійсно є власником секретного ключа.

Бази облікових записів

Один зі способів автентифікації в комп'ютерній системі складається в введенні вашого користувальницького ідентифікатора, у просторіччі називаного «логіном» і пароля – якоїсь конфіденційної інформації, знання якої забезпечує володіння певним ресурсом. Одержавши введений користувачем логін і пароль, комп'ютер порівнює їх зі значенням, що зберігається в спеціальній базі даних і, у випадку збігу, пропускає користувача в систему.

На комп'ютерах з ОС сімейства UNIX, базою є файл /etc/master.passwd (у дистрибутивах Linux звичайно файл /etc/shadow, доступний для читання тільки root), у якому паролі користувачів зберігаються у вигляді хеш функцій від відкритих паролів, крім цього в цьому ж файлі зберігається інформація про права користувача. Споконвічно в Unix – системах пароль (у зашифрованому виді) зберігався у файлі /etc/passwd, доступному для читання всім користувачам, що було небезпечно.

На комп'ютерах з операційною системою Windows 10/11 (не вхідних у домен Windows) така база даних називається SAM (Security Account Manager – Диспетчер захисту облікових записів). База SAM зберігає облікові записи

користувачів, що включають у себе всі дані, необхідні системі захисту для функціонування. Перебуває в директорії %windir%\system32\config\.

У доменах Windows Server 2019 такою базою є Active Directory.

Однак більше надійним способом зберігання автентифікаційних даних визнане використання спеціальних апаратних засобів (компонентів).

При необхідності забезпечення роботи співробітників на різних комп'ютерах (з підтримкою системи безпеки) використовують апаратно-програмні системи, що дозволяють зберігати автентифікаційні дані й криптографічні ключі на сервері організації. Користувачі вільно можуть працювати на будь-якому комп'ютері (робочої станції), маючи доступ до своїх автентифікаційних даних і криптографічних ключів.

Способи автентифікації

Текстове введення логіна й пароля зовсім не є єдиним методом автентифікації. Все більшу популярність набирає автентифікація за допомогою електронних сертифікатів, пластикових карт і біометричних пристроїв, наприклад, сканерів райдужної оболонки ока або відбитків пальців або долоні.

Останнім часом всі частіше застосовується, так звана, розширена або багатофакторна автентифікація. Вона побудована на використанні декількох компонентів, таких як: інформація, що користувач знає (пароль), використанні фізичних компонентів (наприклад, ідентифікаційні брелоки або смарт-карти), і технології ідентифікації особистості (біометричні дані).

Протоколи автентифікації

Процедура автентифікації використовується при обміні інформацією між комп'ютерами, при цьому використовуються досить складні криптографічні протоколи, що забезпечують захист лінії зв'язку від прослуховування або підміни одного з учасників взаємодії. А оскільки, як правило, автентифікація необхідна обом об'єктам, що встановлюють мережну взаємодію, то автентифікація повинна бути взаємною.

Зокрема, в операційних системах сімейства Windows NT 4 використовується протокол NTLM (NT LAN Manager – Диспетчер локальної мережі NT). А в доменах Windows 2019 застосовується набагато більше зроблений протокол Kerberos, що пропонує механізм взаємної автентифікації клієнта й сервера перед установленням зв'язку між ними, причому в протоколі врахований той факт, що початковий обмін інформацією між клієнтом і сервером відбувається в незахищеному середовищі, а передані пакети можуть бути перехоплені й модифіковані. Інакше кажучи, протокол ідеально підходить для застосування в Інтернет і аналогічні мережі.

1.2 Область застосування

Областю застосування розробленого програмного забезпечення, є розподілення доступу до системи на ЕОМ. Широке поширення електронного бізнесу, заснованого на використанні інформаційних технологій, внесло зовсім новий підхід до ведення справ. Основним завданням електронного бізнесу є забезпечення ефективної взаємодії ділових партнерів за рахунок використання сучасних інформаційних технологій. Застосування електронного бізнесу немислимо без наявності розвинених засобів інформаційної безпеки.

Основою будь-якої системи безпеки є контроль доступу, що використовує процедури автентифікації й авторизації для перевірки дійсності користувачів і визначення повноважень доступу до ресурсів, які захищаються. Несанкціонований доступ дійсно завдає серйозної шкоди компаніям, незалежно від того, усвідомлює це керівництво чи ні.

Зрозуміло, що надійність захисту від цієї погрози в першу чергу залежить від якості системи автентифікації користувачів. Сьогодні говорити про інформаційну безпеку без прив'язки до персоналізованого доступу й відстеження всіх дій користувачів у мережі просто не має змісту. Втім, коли мова йде про автентифікації користувачів на комп'ютерах, що входять у корпоративну

локальну мережу, то особливих труднощів не виникає. Ринок пропонує чимало різних рішень, включаючи смарт-карти й електронні ключі, біометричні засоби автентифікації й навіть таку екзотику, як графічні паролі.

Трохи інакше є справи, якщо користувачеві необхідно підключитися до корпоративної комп'ютерної мережі віддаленно, наприклад, через Інтернет. У цьому випадку він може зштовхнутися із цілим рядом проблем, які ми й розглянемо докладніше.

Перебуваючи в недовіреному середовищу (поза офісом), користувач зіштовхується з необхідністю вводити пароль із чужого комп'ютера (наприклад, з Інтернет-кафе). Паролі кешуються, як і будь-яка інша інформація, що вводиться в комп'ютер, і при бажанні ними може скористатися хтось ще у своїх небезкорисливих цілях.

Досить розповсюджений сьогодні й такий вид комп'ютерного шахрайства, як сніффінг (від англ. sniff – нюхати) – перехоплення зловмисником мережних пакетів з метою виявлення його інформації, що цікавить. Використовуючи цей прийом, хакер може "винюхати" пароль користувача й використовувати його для несанкціонованого доступу.

Серйозним випробуванням піддає простий парольний захист (особливо при віддаленому доступі) нове покоління вірусів-шпигунів, що непомітно попадають на комп'ютер користувача в ході звичайного "перелистування" Web-сторінок. Вірус може бути запрограмований на фільтрацію інформаційних потоків конкретного комп'ютера з метою виявлення комбінацій символів, які можуть служити паролями. Ці комбінації "шпигун" пересилає своєму творцеві, а тому залишається тільки виявити потрібний пароль.

Зрозуміло, що апаратний спосіб організації безпечного доступу в мережу в кілька разів надійніше простих паролів, однак як скористатися смарт-картою або USB-ключем, перебуваючи знову ж поза офісом? Швидше за все, це не вдасться, оскільки для першого пристрою потрібний як мінімум зчитувач, для другого – USB-порт, що може бути заблокований (Інтернет-кафе) або, гірше того, його

може попросту не виявитися в пристрої, з якого користувач намагається одержати доступ (КПК, мобільний телефон, смартфон і т.д.). Чи варто навіть говорити про те, що для роботи апаратних засобів – смарт-карт і USB-ключів – необхідно відповідне ПЗ, установити яке в тому же Інтернет-кафе навряд чи можливо.

Тим часом ситуації, коли необхідно віддалено одержати або відправити інформацію, виникають досить часто. Візьмемо хоча б системи електронного банкінга: нескладно уявити собі ситуацію, коли для віддаленого керування своїм рахунком користувачеві буде потрібно доступ до захищених банківських ресурсів. Сьогодні частина банків усвідомила необхідність апаратної авторизації із застосуванням USB-ключа. Але скористатися ним по ряду описаних вище причин можна далеко не завжди.

Специфіка бізнесу багатьох великих компаній часто зобов'язує їх надавати доступ до власних ресурсів стороннім користувачам – партнерам, клієнтам, постачальникам. Сьогодні в Україні активно набирає оберти такий тип співробітництва, як аутсорсінг: компанії-субпідрядникові для виконання робіт на замовлення цілком може знадобитися доступ до захищених ресурсів замовника.

Потреба підключення до корпоративної мережі за надійною схемою автентифікації при наявності під рукою лише КПК або смартфона може стати серйозною проблемою, якщо користувач перебуває на конференції, переговорах або інших ділових заходах. Саме для мобільних додатків, а також для організації доступу до потрібної інформації з тих місць, де неможливо встановити спеціальне ПЗ, була розроблена концепція одноразових паролів OTP – One-Time Password.

Одноразовий пароль – це ключове слово, дійсне тільки для одного процесу автентифікації протягом обмеженого проміжку часу. Такий пароль повністю вирішує проблему можливого перехоплення інформації або банального підглядання. Навіть якщо зловмисник зможе роздобути пароль "жертви", шанси скористатися ним для одержання доступу дорівнюють нулю.

На щастя, сьогодні ситуація змінилася самим кардинальним образом. Загалом кажучи, у західних країнах одноразові паролі для автентифікації саме в інформаційних системах стали звичним явищем. Однак у нашій країні технологія OTP донедавна залишалася недоступною. І лише недавно керівництво компаній початок усвідомлювати, наскільки збільшується ризик несанкціонованого доступу при віддаленій роботі. Попит, як відомо, народжує пропозиція. Тепер продукти, що використовують для віддаленої автентифікації одноразові паролі, стали поступово займати своє місце й на українському ринку.

У сучасних технологіях автентифікації за допомогою OTP застосовується динамічна генерація ключових слів за допомогою сильних криптографічних алгоритмів. Інакше кажучи, автентифікаційні дані – це результат шифрування якого-небудь початкового значення за допомогою секретного ключа користувача. Дана інформація є й у клієнта, і в сервера. Вона не передається по мережі й недоступна для перехоплення. Як початкове значення використовується відома обом сторонам процесу автентифікації інформація, а ключ шифрування створюється для кожного користувача при його ініціалізації в системі

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи проектування сервісів автентифікації, 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

Для забезпечення безпеки інформаційних систем і комп'ютерних мереж застосовується цілий ряд прийомів і технологій, але практично всі вони використовують інформацію про те, хто працює в системі або намагається ввійти в неї. Одержання такої інформації й визначення на її основі, які права буде мати користувач у системі, відбувається в процесі автентифікації.

Сучасні методи автентифікації ґрунтуються на множинних факторах (існує й широко застосовується двофакторна автентифікація, відомі приклади трьохфакторної автентифікації). Наприклад, двофакторна автентифікація користувача проводиться не тільки на основі того, що користувач ЗНАЄ ЩОСЬ (пароль), але й того, що він МАЄ ЩОСЬ (персональний ідентифікатор). Як такий ідентифікатор часто виступає апаратний токен або смарт-карта.

Інтерес до використання токена як засобу апаратної автентифікації з'явився тоді, коли прийшло розуміння, що корпоративна інформація – важливий стратегічний ресурс будь-якого бізнесу. Більше того, цей інтерес відчутний підсилився зі збільшенням мобільності співробітників компаній.

Основні тенденції й сегменти

Засоби для автентифікації користувачів або програмного забезпечення в рамках справжнього огляду – це смарт-карти для IT-безпеки й апаратні токени.

Апаратні токени представлені трьома самостійними сегментами:

– автономні токени;

– USB-токени;

– апаратні пристрої (ключі), службовці для ліцензування й захисту програмного забезпечення (додатків).

На рубежі 2021 і 2022 р. виявилися цікаві й важливі тенденції:

– бізнес усвідомив необхідність управління ідентифікаційними даними й потребу в посиленої автентифікації користувачів;

– стали зрозумілі достоїнства апаратних засобів автентифікації – вони недорогі й можуть використовуватися для роботи в різних мережних середовищах.

Світовий ринок апаратних засобів автентифікації демонструє стійкий ріст. Прогнози авторитетних консалтингових компаній – IDC, Gartner Group – дозволяють припустити, що якщо в 2021 р. дохід від продажів токенів автентифікації склав $318 млн. (приріст на 17,4% у порівнянні з 2020 р.), те до 2022 р. значення цього показника збільшиться до $607 млн.

Таким чином, загальний річний ріст (CAGR) складе 13,8%. Такі ж високі показники демонструє й сегмент смарт-карт.

Автономні токени – це мобільні персональні пристрої, приєднуються не до комп'ютера, які мають власне джерело живлення. Ці пристрої дозволяють користувачеві автентифікувати себе на серверах, використовуючи або одноразовий пароль (токени з використанням OTP – OneTime Password), або метод запит/відповідь.

Суть методу запит/відповідь у тому, що:

– користувач уводить свій ID на робочій станції;

– ID передається по мережі у відкритому виді;

– сервер автентифікації генерує випадковий запит, що передається користувачеві по мережі у відкритому виді;

– користувач уводить запит в автентифікаційний токен:

– токен користувача зашифровує цей запит за допомогою якогось алгоритму шифрування й секретного ключа користувача й результат відображається на екрані;

– користувач уводить результат на робочій станції й ПЗ повертає його серверу;

– сервер зашифровує те ж саме випадкове число (запит);

– при збігу результатів процес запит/відповідь в існуючій системі автентифікації успішно завершується.

USB-токени – пристрою, які підключаються до стандартних портів USB і містять мікроконтролер і/або чип смарт-карти з операційною системою.

USB-токени:

– дозволяють здійснювати строгу двофакторну автентифікацію користувача;

– забезпечують функції шифрування й ЕЦП (цифровий підпис) користувача;

– прямо підключаються до USB-порту комп'ютера (не вимагають зчитувачів);

– не мають потреби в додатковому програмному забезпеченні, установлюваному на сервера (на відміну від OTP-токенів).

В 2021 р. компанія Aladdin випустила комбінований токен, що сполучає функціональність USB- і OTP-токенів.

Електронні ключі для авторизації й ліцензування програмного забезпечення і його захисту від несанкціонованого використання.

Ключі випускаються як для паралельних, так і для USB-портів.

Крім апаратних токенів для автентифікації користувачів широко застосовуються смарт-карти.

Смарт-карти для безпеки – це пластикові картки розміром із кредитку, що містять чип (мікропроцесор) для криптографічних обчислень (ЕЦП, шифрування) і убудовану захищену пам'ять для зберігання інформації (дані про користувача, криптографічні ключі, сертифікати та ін.).

Для використання смарт-карт необхідний зчитувач (карт-рідер).

П'ятірка провідних виробників засобів автентифікації

П'ятірка провідних компаній – постачальників апаратних токенів і їхні частки на світовому ринку за підсумками 2021 р. виглядає так:

– RSA Security – ведучий гравець на ринку апаратних токенів, його частка склала 40,2% світового ринку, а дохід за 2021 рік – 127,8 млн. дол. У порівнянні з 2020 роком прибуток компанії на цьому ринку виросла на 20,8%.

– Aladdin Knowledge Systems і Rainbow Technologies (на початку 2021 р. вона була придбана компанією SafeNet) – володіють рівними частками ринку (близько 15%) і впевнено ділять другі й треті місця. У порівнянні з 2020 роком їхній виторг від продажу токенів збільшилася на 12%, а доходи, отримані компаніями, склали 45,3 і 47,7 млн. дол., відповідно.

– VASCO володіє 5,4% ринку й виручила в 2021 р. 17,2 млн. дол. Зростання прибутку склав 31,7%.

– ActivCard замикає п'ятірку лідерів ринку 2021 року, контролюючи 3,2% ринку й виручивши 10,3 млн. дол. Зростання доходів компанії в порівнянні з 2021 роком склав 43%.

Сегмент автономних токенів

На західному ринку – це самий ємний сегмент на ринку апаратних токенів: прибуток продавців тут склала 177 млн. дол., або 55% загального ринку.

Розподіл часток основних «гравців» на світовому ринку:

– RSA Security (вона контролювала 71,8% ринку й заробила на ньому 127 млн. дол) стала кращим продавцем за результатами 2021 р..

– VASCO посіла друге місце із часток в 9,7% і 17,2 млн. дол. прибутку.

– ActivCard зберегла третє місце, володіючи 5,6% ринку й заробивши 10 млн. дол.

– Secure Computing із прибутком в 6 млн. дол. замикає четвірку гравців цього сегмента.

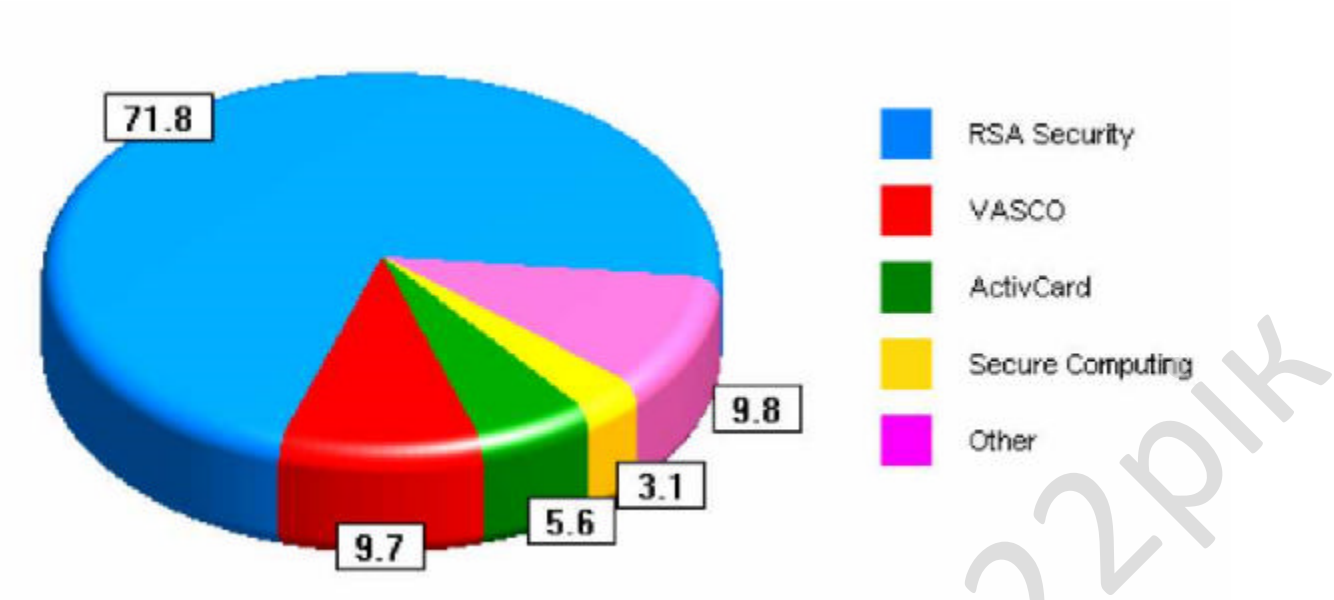

Рисунок 2.1 – Розподіл часток п'ятірки провідних виробників автономних токенів (джерело – IDC, 2021 р.)

Український сегмент ринку автономних токенів дуже слабшав, він почав розвиватися пізніше західного. Поставки OTP-токенів в основному йдуть по українських представництвах великих західних компаній, які вже «сидять» на них і мають у нашій країні свої представництва й філії.

Що стримує застосування OTP-токенів в Україні:

– висока вартість володіння (TCO);

– короткий життєвий цикл OTP-токена (через 3-4 роки «умирає» вбудована батарея й пристрій треба міняти, оплачуючи порядку 70% його початкової вартості).

За даними на кінець березня 2022 р. вартість придбання рішення RSA SecurID для 500 користувачів, що включає:

– основний сервер;

– replicar-сервер (що підключається на час зупинки основного);

– ПЗ RSA Server;

– SecurID ($79 на 1 користувача);

складе $76 000.

Вартість підтримки на рік дорівнює $6,600 за даними дилерів).

Таким чином, вартість рішення складе $82,600, а вартість одного робочого місця, обладнаного OTP-токеном, – не менш $165.

Сегмент USB-токенів

В 2021 році популярність USB-токенів у світі й Україні значно зросла. Приріст цього сегмента світового ринку в порівнянні з 2020 роком склав 67%.

Проте, загальний виторг на цьому сегменті ринку в 2021 р. залишалася щодо невеликий, усього 23 млн. дол.

Основні «гравці»:

– Rainbow Technologies (придбана компанією SafeNet) – лідирує – 29,7% ринку.

– Aladdin посідає друге місце, контролюючи близько 15,5% ринку.

– Gemplus, що зробив ставку на китайський ринок, посів третє місце з 9,6%.

– Eutron – замикає четвірку із часток в 8,7% ринку.

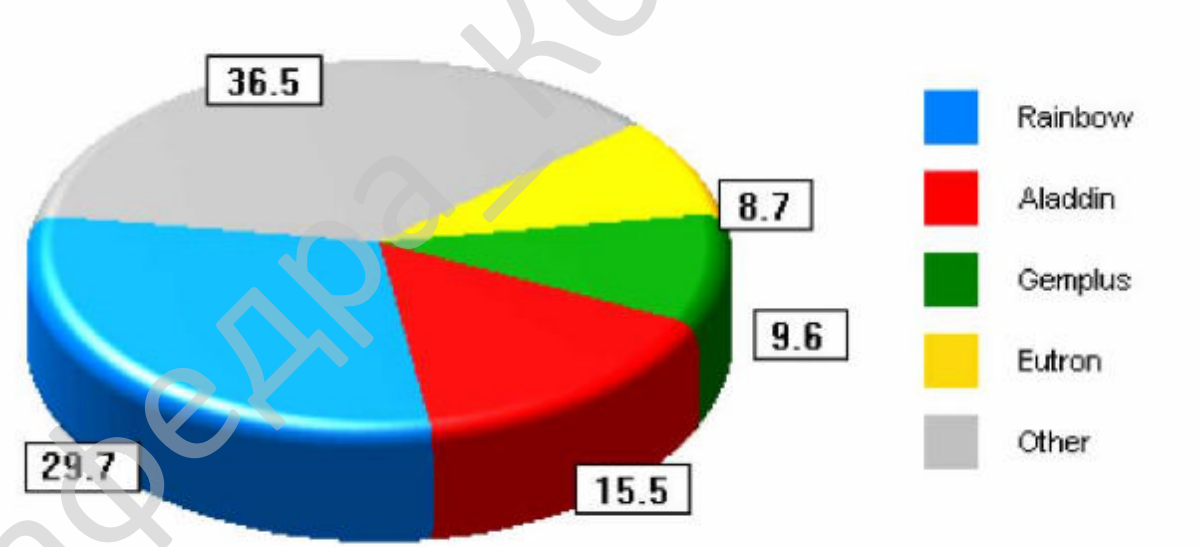

Рисунок 2.2 – Розподіл часток світових виробників USB-токенів

Суть USB-токена – в інтеграції чипа смарт-карти й USB-контролера, що забезпечують комунікацію з комп'ютером і виконують функцією картрідера. Криптографія реалізована в чипі смарт-карти, закриті ключі ніколи не залишають чип, що забезпечує найвищу захищеність цих пристроїв. Цей тип токенів є найбільш затребуваним на ринку й найбільш сумісним з більшістю сучасних додатків.

Інший різновид USB-ключів побудований на базі серійного мікроконтролера, програмно емулюючого функціональність смарт-карти. Такі ключі дешевше й менш захищені.

Обидві технології у свій час були запатентовані компанією Aladdin (патент №6763399 і №6748541).

Український ринок

В Україні, на відміну від західного ринку, частка USB-токенів займає домінуюче положення. Це відбулося завдяки успішній маркетинговій політиці й агресивному просуванню саме USB-ключів на ринок, що формується.

Частка USB-токенів на українському ринку щодо інших видів токенів оцінюється приблизно в 80-85%. При цьому сильно домінують токени західного виробництва – їхня частка становить близько 90%.

Основні «гравці»:

– Aladdin займає лідируюче положення, контролюючи близько 70% ринку.

– Rainbow Technologies (дистриб'ютор компанії SafeNet) – на другому місці – близько 25% ринку.

– Актив (разом з компанією Анкад, виробником криптографії) – український розроблювач і виробник ключів – на третім місці із часток порядку 5% ринку.

Частки інших «гравців» (як правило, дилерів західних компаній-виробників) мізерно малі.

Rainbow Technologies – український дистриб'ютор компанії SafeNet. Продукт – USB-ключі iKey.

Ключі iKey в Україні не сертифіковані.

Компанія поставляє на ринок тільки USB-ключі:

– iKey-1000 – на базі серійного мікроконтролера с програмною емуляцією функціональності смарт-карти;

– iKey 2000 і 3000 на базі смарт-карт (різних виробників, і, як наслідок, що мають несумісні між собою драйвери й ПЗ).

Актив – український розроблювач і виробник USB-ключів ruToken. Ключі ruToken не сертифіковані. Ліцензій на експорт ключів немає. ruToken побудований на базі серійного мікроконтролера, у якому апаратно реалізований тільки український симетричний ДСТ 28147-89, апаратної підтримки західних алгоритмів немає – при їхньому виклику здійснюється «проброс» до стандартному криптопровайдеру. Ключ у цьому випадку виконує функцію «захищеної PIN кодом флеш-пам'яті» для зберігання ключових контейнерів.

Основні тенденції ринку

– Інтерес до проблеми автентифікації в Україні підігрівається «дорослішанням» української законодавчої бази й прийняттям законів «Про електронний цифровий підпис», «Про захист авторських прав» і закону «Про комерційну таємницю».

– Користувальницькі переваги перебувають в області західних засобів як більше надійних, сумісних і технологічно зроблених.

– Помітне пожвавлення на українському ринку токенів відбувся влітку 2021 р. – ринок дозрів і остаточно «прокинувся». Багато проектів, початі як пилотніе рік-два назад, перейшли у фазу промислового впровадження.

– Спостерігається помітний інтерес до USB-токенам з боку державних установ, але поставляємі їм токени зобов'язані мати відповідний сертифікат.

– Вітчизняна продукція відстає й у технологічному, і в технічному плані. Претендувати на сегмент масового ринку українські розробки поки не можуть, їхня ніша – ринок спецдодатків.

Апаратні пристрої для ліцензування й захисту ПЗ

В 2021 році на ринку USB-ключів спостерігався стриманий ріст, багато компаній використовували їх для боротьби з комп'ютерним піратством.

У порівнянні з 2020 роком загальний виторг даного сегмента ринку виросла на 14,5%, склавши 117,8 млн дол.

Основні гравці цього сегмента світового ринку:

– Aladdin Knowledge Systems залишалася лідером сегмента, контролюючи 35,4% ринку. Rainbow Technologies – на другому місці (35,1% ринку).

– WIBU-Systems, відома німецька компанія, була третім гравцем, що переборов бар'єр в 5% ринку (6,6%).

Ринок смарт-карт для IT-безпеки

У звіті Burton Group (08.2021 р.) серйозна увага приділена сегменту багатофункціональних смарт-карт для IT-безпеки (далі ми будемо говорити тільки про цей сегмент).

Смарт-карта для IT-безпеки – персональний засіб автентифікації й зберігання даних (наприклад, відкритого ключа для технології PKI), вона апаратно підтримує роботу із цифровими сертифікатами й ЕЦП, має убудовану криптографію для формування ЕЦП і виконання криптографічних операцій, захищену пам'ять для зберігання ключів і сертифікатів.

Раніше стримуючим фактором розвитку сегмента смарт-карт була відносно висока вартість рідерів. В 2020-2021 г. вартість рідерів сильно знизилася (до 25-45$). Комплект «рідер + смарт-карта» став продаватися за ціною USB-токена. Ринок відгукнувся негайним збільшенням попиту на цю продукцію.

Помітною тенденцією сегмента смарт-карт для інформаційної безпеки стало те, що великі західні виробники (Gemplus, Schlumberger Company/Axalto) пішли в область банківських карток і SIM-модулів для стільникових телефонів. Потреба ринку в смарт-картах велика, але провідні вендори перемкнулися на діяльність, що зараз їм приносять більші гроші. Ніша, що утворилася, скоріше не заповнена.

Україна: в 2021 році Aladdin оголосив про повернення на сегмент смарт-карт і випустив на ринок e-Token у вигляді смарт-карт. Тепер, не міняючи ПЗ, користувач зможе користуватися токеном у кожному із двох форм-факторів. Пристрій був сертифікований і може рекомендуватися для використання в держстуктурах. Це сильний хід, що забезпечив компанії першість.

Другий гравець українського сегмента – Axalto – його частка вкрай незначна й позиція пасивна.

Третій – РІК (українська інтелектуальна карта). Технологічна недосконаліть карти визначило її долю. Ринок перебуває чекаючи продукту РІК2 (виробник – АНГСТРЕМ) – спеціалізованого продукту в області безпеки, але очікування затяглося.

Інших помітних гравців на ринку смарт-карт для ІТ-безпеки в Україні немає.

Розвиток світового ринку

Аналітики впевнені, що темпи впровадження систем багатофакторної автентифікації згодом будуть тільки зростати. Аналізуючи ймовірні стратегії споживачів і динаміку сегментів ринку апаратних засобів автентифікації (токенів), вони затверджують, що світовий ринок буде формуватися під впливом наступних факторів:

Паролі не безкоштовні. Дослідження Gartner показують, що від 10 до 30% дзвінків у службу технічної підтримки компанії – прохання співробітників із приводу відновлення забутих ними паролів.

По даним IDC кожний забутий пароль обходиться організації в 10 – 25 доларів.

Токени рятують від необхідності запам'ятовування численних паролів, знімаючи цю проблему.

Компанії зацікавлені в спрощенні процедур автентифікації при доступі до ресурсів і скороченні кількості користувальницьких паролів. Технологія

однократної автентифікації дозволяє надати користувачам доступ до додатків і інформаційних ресурсів компанії.

Компанії відкривають доступ до своїх мереж все більшій кількості віддалених користувачів (власним співробітникам, партнерам, замовникам). Як правило, захист подібного доступу здійснюється за допомогою технології VPN, заснованої на протоколах IPSec або SSL/TLS. VPN забезпечує надійне шифрування переданих даних і автентифікацію сторін інформаційного обміну. Однак у рамках процедури автентифікації нерідко використовуються звичайні паролі, що не забезпечують належного рівня довіри. З огляду на це, багато компаній всі частіше вдаються до допомоги апаратних засобів автентифікації, серед яких найбільш популярні USB-токени.

Основні виробники ПЗ все частіше вбудовують підтримку токенів до складу розроблювальних ними продуктів. У рамках корпоративних додатків токени можуть бути використані для автентифікації користувачів, зберігання криптографічних ключів, формування електронного цифрового підпису й т.д.. Додатки, розроблені компаніями Microsoft, Cisco Systems, Novell, RSA Security та ін. пропонують різні можливості застосування токенів. Компанія Verisign, що є світовий постачальник рішень в області автентифікації пропонує використовувати апаратні пристрої нового покоління, що сполучають функціонали OTP- і USB-токенів (дані пристрої виконані на базі ключів eToken NG OTP компанії Aladdin Knowledge Systems).

Існують пристрої, що у строгому розумінні не є апаратними токенами. Однак вони можуть застосовуватися для строгої автентифікації користувачів. Основні тенденції:

– Пристрій USB-flash – дуже популярний продукт, що поширюється усе ширше. Сьогодні ці пристрої використовуються тільки для зберігання даних, але в найближчому майбутньому вони одержать функції безпеки й зможуть бути використані для автентифікації. Компанії KeyComputing і M-Systems, приміром, працюють у цьому напрямку.

– Ще одна технологія, що може вплинути на розвиток ринку – це використання SIM-карт, активно застосовуваних сьогодні в мережах стільникового зв'язка. SIM-карти можуть використовуватися й для автентифікації користувачів при доступі до корпоративних ресурсів. У деяких додатках, наприклад у рамках бездротових локальних мереж (WLAN) використання SIM-карти мобільного телефону може бути підходящої автентифікаційним рішенням. Створення SIM-карт із дуальним інтерфейсом послужить гарним імпульсом для подібної реалізації.

Об'єднання функціональних можливостей різних типів токенів уважається перспективною тематикою. Ключі для ліцензування й захисту ПЗ поступово переходять від використання паралельних і серійних портів до технології USB. Сьогодні активно вдосконалюються можливості USB-токенів автентифікації. Розвивається лінійка токенів, які можуть бути використані і як пристрої для контролю фізичного доступу, наприклад, продемонстровані SafeNet токени RfiKey, eToken RM компанії Aladdin, або токени з інтегрованими зчитувачами біометричних даних, як Puppy від Sony. Крім того, ActivCard, Aladdin і Authenex оголосили про випуск гібридних токенів, які будуть поєднувати можливості традиційних (використовуючих OTP) і USB-токенів.

Постачальники ПЗ зацікавлені в можливості переконатися в тому, що їх ПЗ використовується відповідно до заданої їх політикою ліцензування. ПЗ стає модульним, при цьому покупцям надаються повні набори програм, але набір можливостей буде задаватися ліцензіями, що зберігаються в ключах. Для комплексного контролю програм можуть бути використані ключі для ліцензування й захисту ПЗ. У рішенні цієї проблеми, на думку аналітиків IDC, відбудеться міграція переваг убік технології USB.

Виводи аналітиків

Основними гравцями західного ринку апаратних токенів залишаться гравці нинішні, появи нових суперників в Україні й світі не відбудеться, а можливість консолідації примарна.

Аналітики багатьох консалтингових компаній, у тому числі IDC і Gartner Group, уважають, що до 2022 р. застосування USB-токенів автентифікації виросте до того рівня, якого досягло використання OTP-токенів. Gartner Group називає USB-токени й смарт-карти для інформаційної безпеки кращим інвестиційним вкладенням у забезпечення безпечного доступу до даних в 2022 році.

В Україні відбудеться впорядкування ринку засобів автентифікації. Це зв'язано:

– с усвідомленням цінності корпоративної інформації;

– с підвищенням інтересу держструктур до засобів, що забезпечують строгу автентифікацію користувачів, можливість застосування ЕЦП і управління доступом;

– с прийняттям ряду нових законів;

– с упорядкуванням роботи митної служби.

Компанії, які пропонують неліцензовані або несертифіковані засоби безпечного доступу, «сірі» постачальники й т.п. будуть витиснуті з ринку.

У фокусі споживчих переваг будуть ті українські постачальники, які мають найбільш чітку, сфальцьовану політику, продуману лінійку продуктів і систему управління токенами.

eToken Алладин

eToken – персональний засіб строгої автентифікації й зберігання даних, апаратно підтримуючу роботу із цифровими сертифікатами й ЕЦП. eToken випускається у двох форм-факторах: USB-ключа й смарт-карти. eToken підтримує роботу й інтегрується з усіма основними системами й додатками, що використовують технології смарт-карт або PKI (Public Key Infrastructure). eToken може виступати в якості єдиної корпоративної карти, що служить для візуальної ідентифікації співробітника, для доступу в приміщення, для входу в комп'ютер, у мережу, для доступу до захищених даних, для захисту електронних документів (ЕЦП, шифрування), для встановлення захищених з'єднань (VPN, SSL), для проведення фінансових транзакцій.

Сертифіковані моделі:

– eToken PRO (Java) – Нове покоління USB-ключів і смарт-карт eToken, побудоване на базі Java-карти. eToken PRO (Java) має всю функціональність eToken PRO, має збільшений обсяг пам'яті для захищеного зберігання користувальницьких даних (72КБ) і надає можливість розширення функціонала за рахунок завантаження додаткових додатків (аплетів). Випускається у вигляді моделей: USB-ключ і смарт-карта.

– eToken PRO – eToken PRO являє собою захищений пристрій, призначений для строгої автентифікації, безпечного зберігання секретних даних, виконання криптографічних обчислень і роботи з асиметричними ключами й цифровими сертифікатами. Випускається у вигляді моделей: USB-ключ і смарт-карта.

– eToken ДСТ – eToken ДСТ – це персональний засіб формування ЕЦП із невитягаємим закритим ключом. eToken ДСТ призначений розроблювачам систем дистанційного банківського обслуговування, електронних торгів, здачі податкової звітності для вбудовування в клієнтську частину, а також розроблювачам СКЗІ для забезпечення надійного захисту закритих ключів і аппартной реалізації ЕЦП. Для вбудовування eToken ДСТ у додатки власної розробки варто використовувати комплект розроблювача eToken ДСТ – eToken ДСТ SDK. eToken ДСТ перебуває на сертифікаційних випробуваннях як засіб криптографічного захисту інформації з рівня КС2.

– eToken NG-FLASH – eToken NG-FLASH – комбінований ключ для використання в системах інформаційної безпеки, що сполучить можливості смарт-карти й захищеного сховища даних.

– eToken NG-OTP – eToken NG-OTP – це перший USB-ключ для інформаційної безпеки, заснований на мікросхемі смарт-карти, з генератором одноразових паролів (OTP – One Time Password).

– eToken PASS – eToken PASS – автономний генератор одноразових паролів. Для централізованого управління пристроями в масштабах підприємства

використовується система eToken TMS. Пристрій не вимагає підключення до комп'ютера й установки додаткового ПЗ, і тому може використовуватися в будь-яких операційних системах, а також при доступі до захищених ресурсів з мобільних пристроїв і терміналів, що не мають USB-роз’єм.

– eToken з радіо-міткою RFID – eToken з убудованої в нього радіо-міткою RFID використовується для інтеграції із системами контролю й управління доступом у приміщення (СКУД), безконтактних "електронних прохідних", системах обліку робочого часу персоналу й т.д.

Розглянемо більш докладно один із представлених продуктів.

eToken PRO

– Виконаний на базі мікросхеми смарт-карти.

– Апаратно реалізовані алгоритми RSA 1024/2048, DES/56, TripleDES/168, SHA-1, MAC, iMAC.

– Апаратна генерація ключових пар RSA 1024/2048.

– Найвищий рівень безпеки.

– Захищена пам'ять обсягом до 64 КБ на мікросхемі смарт-карти.

– Моделі: USB-ключ і смарт-карта.

– Може вбудовуватися радіо-мітка (RFID).

eToken PRO являє собою захищений пристрій, призначений для строгої автентифікації, безпечного зберігання секретних даних, виконання криптографічних обчислень і роботи з асиметричними ключами й цифровими сертифікатами.

eToken PRO рекомендується для строгої двофакторної автентифікації з використанням сертифікатів відкритих ключів, а також:

– для використання в PKI-додатках;

– у корпоративних проектах;

– у додатках, що підтримує технології смарт-карт;

– у додатках електронної комерції;

– у банківських додатках.

USB-ключ eToken PRO архітектурно реалізований як USB карт-рідер з убудованої в нього мікросхемою (чипом) смарт-карти. Ключ виконаний у вигляді брелока й прямо підключається до порту USB, при цьому не вимагає для своєї роботи яких-небудь додаткових пристроїв.

Смарт-карта eToken PRO може використовуватися з будь-яким стандартним PC/SC сумісним рідером. Функціонально USB-ключ і смарт-карта eToken PRO ідентичні й виконані на одній і тій же мікросхемі смарт-карти; вони однаково підтримуються їхніми додатками, що використовують.

eToken PRO випускається у двох модифікаціях: з розміром пам'яті 32 КБ і 64 КБ.

Інтеграція із системами контролю доступу. Смарт-карти й USB-ключі eToken PRO на замовлення можуть випускатися з убудованими пасивними радіо-мітками RFID для контролю фізичного доступу в приміщення й контролю логічного доступу до інформаційних ресурсів. Застосовуються в системах контролю й управління доступом (СКУД), безконтактних "електронних прохідних", системах обліку робочого часу персоналу й т.п.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних

багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++

використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 проектування сервісів автентифікації.

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

Концепція одноразових паролів у системі автентифікації на основі USB-токена

Технології OTP були розроблені в рамках галузевої ініціативи Open Authentication (OATH), висунутою компанією VeriSign в 2004 році. Суть цієї ініціативи полягає в розробці стандартної специфікації дійсно надійної автентифікації для різних сервісів. Причому мова йде про двофакторне визначення прав користувачів, у процесі якого останній повинен "пред'явити" смарт-карту або USB-токен і свій пароль. Таким чином, одноразові паролі згодом можуть стати стандартним засобом віддаленої автентифікації в різних системах.

Сьогодні розроблено й використовується на практиці кілька методів реалізації систем автентифікації по одноразових паролях.

Метод "запит-відповідь"

Принцип його роботи такий: на початку процедури автентифікації користувач відправляє на сервер свій логін. У відповідь на це останній генерує якийсь випадковий рядок і посилає її назад. Користувач за допомогою свого ключа зашифровує ці дані й відправляє їх назад. Сервер же в цей час "знаходить" у своїй пам'яті секретний ключ даного користувача й кодує з його допомогою вихідний рядок. Далі проводиться порівняння обох результатів шифрування. При їхньому повному збігу вважається, що автентифікація пройшла успішно. Цей метод реалізації технології одноразових паролів, на відміну від всіх інших, називається асинхронним, оскільки процес автентифікації не залежить від історії роботи користувача із сервером і іншими факторами.

Метод "тільки відповідь"

При його використанні алгоритм автентифікації виглядає трохи простіше. На самому початку процесу програмне або апаратне забезпечення користувача самостійно генерує вихідні дані, які будуть зашифровані й відправлені на сервер для порівняння. При цьому в процесі створення рядка використовується значення попереднього запиту. Сервер теж має такими "знання". Тобто він, використовуючи ім'я користувача, знаходить значення попереднього його запиту й генерує по тому же алгоритму точно такий же рядок. Зашифрувавши його за допомогою секретного ключа користувача (він також зберігається на сервері), сервер одержує значення, що повинне повністю збігатися із присланими користувачем даними.

Метод "синхронізація за часом"

У ньому як вихідний рядок виступають поточні показання таймера спеціального пристрою або комп'ютера, на якому працює людина. При цьому звичайно використовується не точна вказівка часу, а поточний інтервал із установленими заздалегідь границями (наприклад, 30 секунд). Ці дані зашифровуються за допомогою секретного ключа й у відкритому виді відправляються на сервер разом з ім'ям користувача. Сервер при одержанні запиту на автентифікацію здійснює ті ж дії: одержує поточний час від свого таймера й зашифровує його. Після цього йому залишається тільки зрівняти два значення: обчислене й отримане від віддаленого комп'ютера.

Метод "синхронізація по події"

У принципі, цей метод практично повністю ідентичний попередньій розглянутій технології. От тільки як вихідний рядок у ньому використовується не час, а кількість успішних процедур автентифікації, проведених до поточної. Це значення підраховується обома сторонами окремо друг від друга. В іншому ж даний метод повністю ідентичний попередній.

У деяких системах реалізуються так звані змішані методи, де як початкове значення використовується два або навіть більше типи інформації. Наприклад,

існують системи, які враховують як лічильники автентифікацій, так і показання убудованих таймерів. Такий підхід дозволяє уникнути безлічі недоліків окремих методів.

Уразливості технологій OTP

Технологія одноразових паролів вважається досить надійною. Однак об'єктивності заради відзначимо, що й у неї є свої недоліки, яким піддані всі системи, що реалізують принцип OTP у чистому виді. Подібні уразливості можна розділити на дві групи. До першого ставляться потенційно небезпечні "діри", властивим всім методам реалізації. Найбільш серйозної з них є можливість підміни сервера автентифікації. При цьому користувач буде відправляти свої дані прямо зловмисникові. Ну а він може відразу використовувати їх для доступу до справжнього сервера. У випадку застосування методу "Запит-Відповідь" алгоритм атаки небагато ускладнюється (комп'ютер хакера повинен зіграти роль "посередника", пропускаючи через себе процес обміну інформацією між сервером і клієнтом). Втім, варто відзначити, що на практиці здійснити таку атаку зовсім непросто.

Інша уразливість властива тільки синхронним методам реалізації одноразових паролів, оскільки існує ризик розсинхронізації інформації на сервері й у програмному або апаратному забезпеченні користувача. Допустимо, у якійсь системі початковими даними є показання внутрішніх таймерів. І з якихось причин вони починають не збігатися один з одним. У цьому випадку всі спроби користувачів пройти автентифікацію будуть невдалими (помилка першого роду). На щастя, у подібних випадках помилка другого роду (помилковий допуск "чужого") виникнути не може. Втім, варто відзначити, що ймовірність виникнення описаної ситуації також украй мала.

Деякі атаки застосовні тільки до окремих способів реалізації технології одноразових паролів. Для приклада можна знову взяти метод синхронізації по таймері. Як ми вже говорили, час у ньому враховується не з точністю до секунди, а в межах якогось установленого заздалегідь інтервалу. Це необхідно для обліку

можливості розсинхронізації таймерів, а також появи затримок у передачі даних. І саме цим моментом теоретично може скористатися зловмисник для одержання несанкціонованого доступу до віддаленої системи. Для початку хакер "прослуховує" мережний трафік від користувача до сервера автентифікації й перехоплює відправлені "жертвою" логін і одноразовий пароль. Потім він відразу блокує його комп'ютер (перевантажує його, обриває зв'язок і т.п.), а сам відправляє авторизаційні дані вже від себе. І якщо він встигне зробити це так швидко, щоб інтервал автентифікації не встиг змінитися, то сервер визнає його за зареєстрованого користувача.

Зрозуміло, що для такої атаки зловмисник повинен мати можливість "прослуховування" трафіка, а також швидкого блокування комп'ютера клієнта, а реально здійснити це – завдання не з легких. Найпростіше виконати ці умови тоді, коли атака замислюється заздалегідь, причому для підключення до віддаленої системи "жертва" буде використовувати комп'ютер із чужої локальної мережі. У цьому випадку хакер може заздалегідь "попрацювати" над одним із ПК, одержавши можливість управляти їм з іншої машини. Захиститися від такої атаки можна тільки шляхом використання "довірених" робочих машин (наприклад, власний ноутбук або КПК) і "незалежних" захищених, наприклад, за допомогою SSL, каналів виходу в Інтернет.

Якість реалізації

Надійність будь-якої системи безпеки в значній мірі залежить від якості її реалізації. Тобто в практичних рішень є свої недоліки, які можуть бути використані зловмисниками у своїх цілях. Причому ці "діри" найчастіше не ставляться прямо до реалізованої технології. Повною мірою це правило застосовне й до систем автентифікації на базі одноразових паролів. Як уже говорилося вище, у їхній основі лежить використання криптографічних алгоритмів. Це накладає певні зобов'язання на розроблювачів таких продуктів. Адже неякісне виконання якого-небудь алгоритму або, наприклад, генератора випадкових чисел, може поставити під погрозу безпека інформації.

Генератори одноразових паролів можуть бути реалізовані двома способами: програмним і апаратним. Перший з них, природно, менш надійний. Справа в тому, що клієнтська утиліта повинна зберігати в собі секретний ключ користувача. Зробити це більш-менш безпечно можна тільки за допомогою шифрування самого ключа на основі персонального пароля. При цьому необхідно враховувати, що клієнтська утиліта повинна бути встановлена на комп'ютері (КПК, смартфон і т.п.), з якого в цей момент людин здійснює сесію. Таким чином, виходить, що автентифікація співробітника залежить від одного пароля. Тим часом існує безліч способів довідатися або підібрати його. І це далеко не єдина уразливість програмного генератора одноразових паролів.

Незрівнянно більшою надійністю володіє широкий спектр пристроїв для апаратної реалізації OTP-технологій. Наприклад, є пристрої, по виду нагадують калькулятор: при введенні в них набору цифр, присланого сервером, на основі вшитого секретного ключа вони генерують одноразовий пароль (реалізація методу "Запит-Відповідь"). Головна уразливість подібних пристроїв полягає в можливості його втрати або злодійства. Якщо ваш "калькулятор" потрапить у руки зловмисникові, то останній зможе одержати бажаний доступ до віддаленої системи. Убезпечити систему від подібних дій можна тільки за умови використання надійного захисту пам'яті пристрою із секретним ключем.

Саме такий підхід реалізований у смарт-картах і USB-токенах. Для доступу до їхньої пам'яті користувач повинен увести свій PIN-код. Причому дані пристрої захищені від його підбора: при трикратному введенні неправильного значення вони блокуються. Надійне зберігання ключової інформації, апаратна генерація ключових пар і виконання криптографічних операцій у довіреному середовищі (на чипі смарт-карти) унеможливлюють спроби зловмисника витягти секретний ключ і виготовити дублікат пристрою генерації одноразових паролів.

3.2 Розробка структурної схеми

Спершу розглянемо питання розробки методології формування процесу автентифікації. Найважливішою частиною підсистеми автентифікації є сукупність алгоритмів автентифікації, які задають набір захисних функцій, що визначають, що й при яких умовах може бути захищено.

Таблиця 3.1 – Упорядкованість захисних функцій автентифікації

Координати захисних функцій	Упорядкованість
Тип автентифікації	автентифікація повідомлення ⊇ упізнавання
Число сеансів на одному ключі	багаторазова автентифікація ⊇ однократна автентифікація
Тип використовуваного каналу зв'язку (можливість діалогу)	бездіалогова автентифікація ⊇ діалогова автентифікація
Довіра до верифікатору	недоверенный верифікатор ⊇ довірений верифікатор
Якість зв'язку (при однократної автентифікації потрібне надійне доведення інформації)	некритичність надійного доведення інформації ⊇ необхідність надійного доведення інформації
Наявність служби єдиного часу (необхідно для захисту від повторів або затримок інформації при бездіалогової автентифікації)	необов'язковість єдиного часу ⊇ наявність єдиного часу
Відносний обсяг переданої службової інформації (відношення обсягу переданих даних до ентропії	менший відносний обсяг службової інформації ⊇ більший відносний обсяг службової інформації

У роботі до складу алгоритмів автентифікації включені:

– власно протоколи автентифікації;

– швидкі алгоритми, що реалізують обчислення у відповідних математичних структурах;

– допоміжні алгоритми, що впливають на безпеку;

– алгоритми вибору параметрів підсистеми автентифікації;

– алгоритми керування ключами, включаючи зміну параметрів підсистеми автентифікації.

Стандартні алгоритми автентифікації не завжди задовольняють вимогам, пропонованим до критичних інформаційно-телекомунікаційних систем, тому з'являється необхідність проектування оригінальних алгоритмів автентифікації. Безпека цих алгоритмів традиційно ґрунтується на складності рішення математичного завдання, для вибору якого в ході проектування пропонується трьохрівневасистематизація завдань.

До першого рівня віднесені класи уніфікованих математичних завдань (КУМЗ), у якості яких запропоноване розглядати наступні типи завдань, орієнтованих на побудову алгоритмів автентифікації:

– завдання про виконуваність (до якої зводяться завдання розкриття ключа, обіги й обчислення колізій хеш-функції);

– завдання визначення структури й порядку кінцевої групи;

– завдання обчислення індексу елемента кінцевої абелевої групи;

– завдання про укладання ранця;

– завдання обчислення морфізма між об'єктами категорії.

До другого рівня віднесені масові основні математичні завдання вибору (ОМЗ), отримані в результаті параметризації КУМЗ за допомогою математичних структур, що визначають область математики, до якої відноситься ОМЗ, а також класи зв'язаних завдань, до яких зводиться ОМЗ. Крім завдань вибору при дослідженні безпеки використовуються також додаткові завдання розпізнавання й пошуку.

До третього рівня віднесені приватні математичні завдання, що відповідають масовій ОМЗ.

Проектування алгоритмів автентифікації (рисунок 3.1) пропонується формально визначати як процес побудови ланцюжків відображень:

{Класи уніфікованих математичних завдань} ×
× {уніфіковані криптографічні примітиви} →
→{мінімізований набір захисних функцій} →
→ {узагальнені протоколи автентифікації};

{Класи уніфікованих математичних завдань} ×
× {математичні структури} →
→ {Основні математичні завдання} ∪ {додаткові завдання};

{Основні математичні завдання} × {узагальнені протоколи автентифікації} →
→ {алгоритми автентифікації} →
→ {швидкі обчислювальні алгоритми};

{Основні математичні завдання} ∪ {додаткові завдання} →
→ {алгоритми генерації параметрів підсистеми автентифікації} →
→ {частки математичні завдання} →
→ {алгоритми керування ключами}.

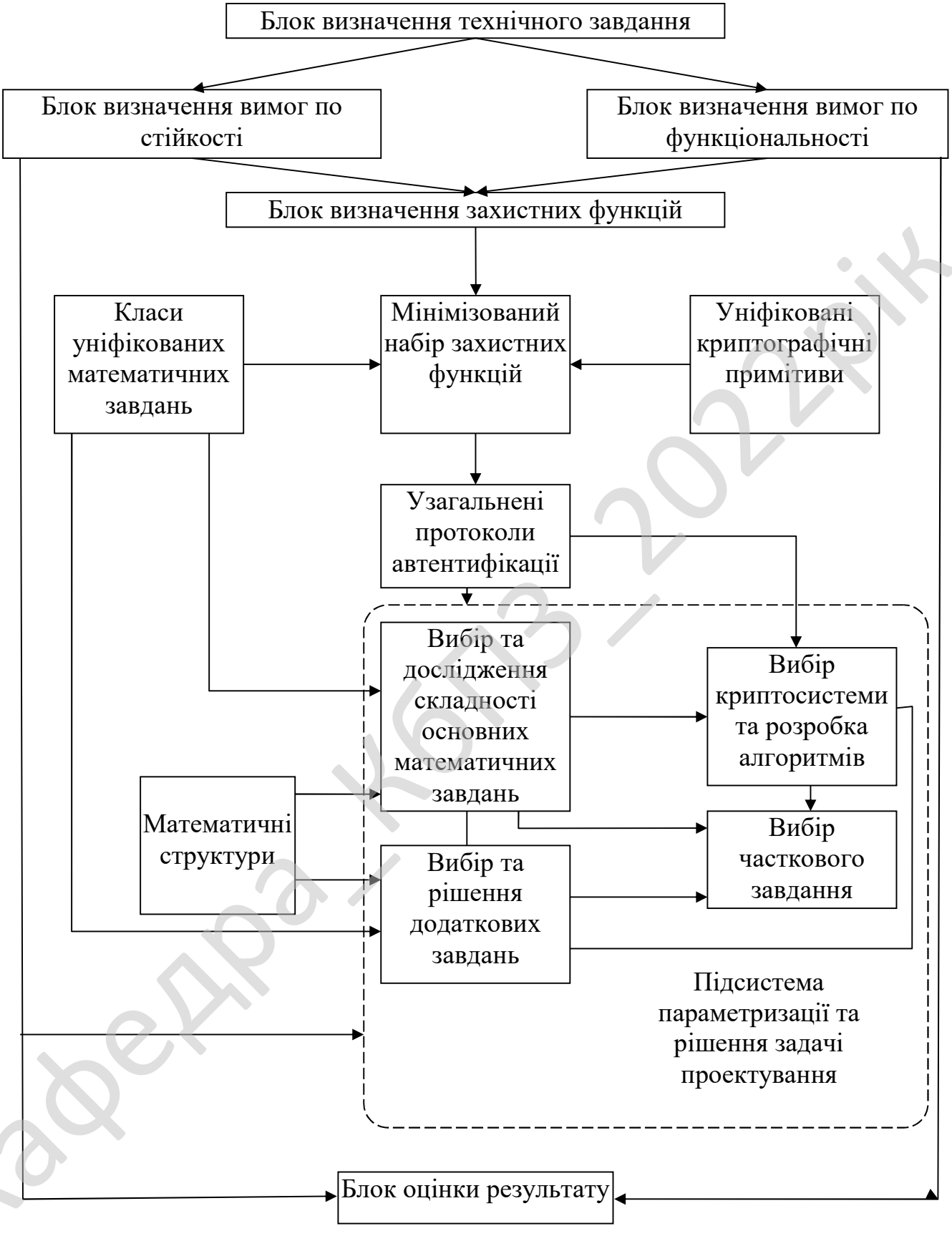

Рисунок 3.1 – Структурна схема системи проектування алгоритмів автентифікації

Уніфіковані криптографічні примітиви містять у собі: симетричне шифрування, шифрування з відкритим ключем, безключову й ключову хеш-функцію, цифровий підпис, діалогові й бездіалогові докази з нульовим розголошенням знань, секретні гомоморфізми. Безлічі математичних завдань і криптографічних примітивів варто розглядати в їхньому розвитку.

При проектуванні алгоритмів автентифікації потрібно прогнозувати як зниження складності математичного завдання, так і ріст продуктивності обчислювальної техніки, що дозволяє вирішити це завдання, а також розвиток інших (не зв'язаних безпосередньо з обчисленнями) можливостей порушника, спрямованих на зниження безпеки.

Швидкість $s(t,T)$ падіння стійкості $S(t)$ на інтервалі часу $(T,\ T+t)$ запропоновано визначати по формулі:

$$s(t, T) = (\log S(T) - \log S(T + t))/(t \log S(T)).$$

Показано, що складність завдань падає приблизно з постійною швидкістю. Отримані оцінки дозволяють прогнозувати зниження складності й визначати час життя ключа.

Завдання, покладені в основу безпеки алгоритмів автентифікації, пропонується класифікувати по трьох типах: вибір, розпізнавання, пошук. У ході проектування звичайно потрібно обґрунтувати складність завдання вибору й знайти або оцінити рішення завдань розпізнавання й пошуку.

Математичні структури, використовувані при проектуванні алгоритмів автентифікації в умовах довіреного верифікатора, як правило, не мають ефективно розв'язні алгебраїчні властивості й розрахункові статистичні характеристики. Це викликає необхідність введення додаткових завдань і обумовлює їхню складність.

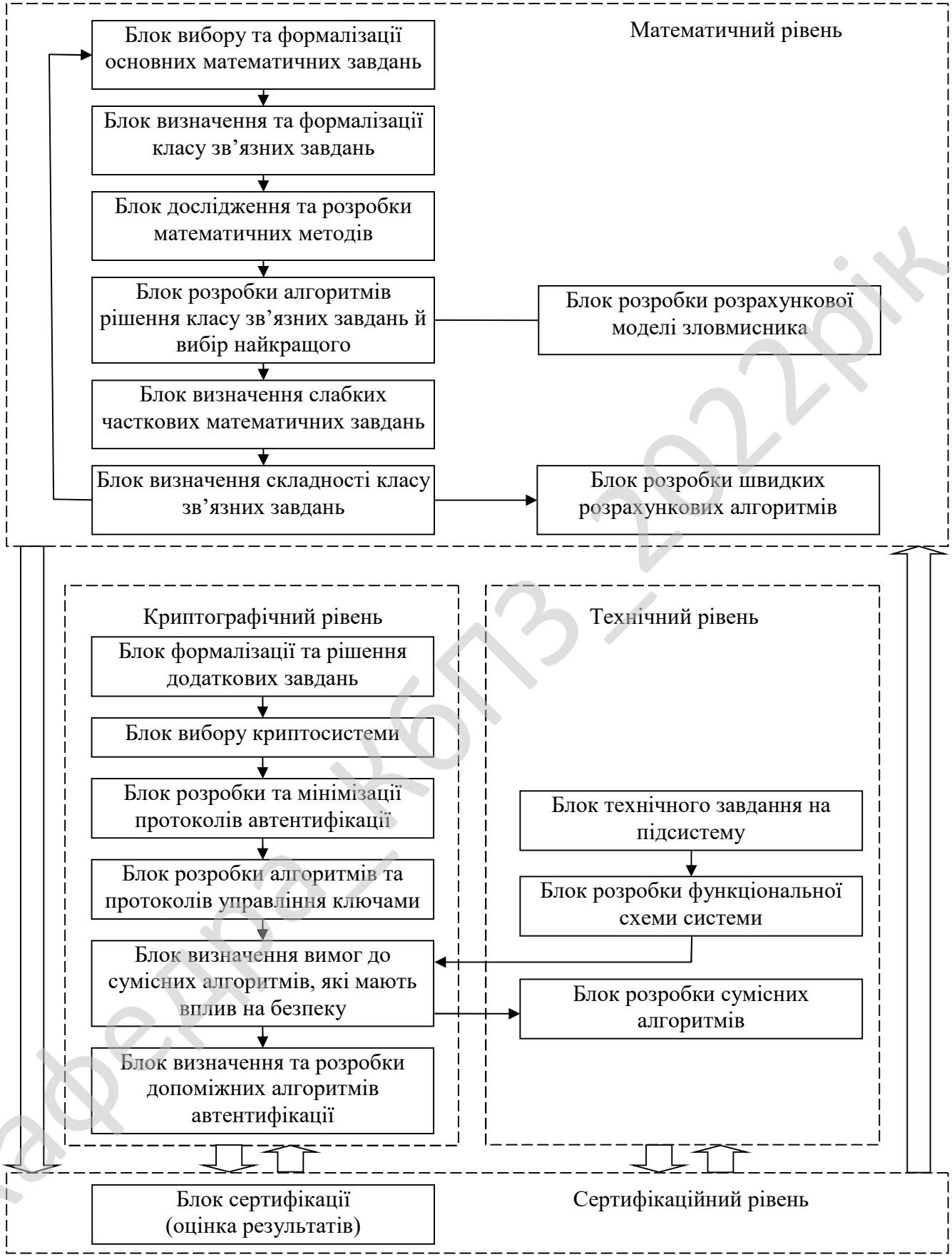

Рисунок 3.2 – Структурна схема системи параметризації та рішення завдання проектування

Для автентифікації в умовах недовіренного верифікатора використовуються математичні структури з розпізнаваними алгебраїчними властивостями (групи, кільця, категорії) і відповідні ОМЗ.

На підставі аналізу особливостей застосування ОМЗ сформульовані вимоги до ОМЗ і відповідних класів зв'язаних завдань (КЗЗ), до яких поліномиально зводиться завдання порушення безпеки.

Для параметризації й рішення завдання проектування алгоритмів автентифікації пропонується чотирьохрівнева схема (рисунок 3.2), що характеризується, крім традиційного технічного рівня, наявністю математичного, криптографічного й сертифікаційного рівнів.

Математичний рівень є найбільш специфічним, у значній мірі визначає трудомісткість проектування, вимагає високої наукової кваліфікації виконавців і із цієї причини не може бути ефективно розпараллелений.

Розглянувши запропоновану методологію побудови систем автентифікації перейдемо до практичної побудови системи автентифікації з використанням наведених вище теоретичних відомостей.

За основу системи автентифікації візьмемо автентифікацію з одноразовим паролем з застосуванням USB-ключа.

Структурна схема такої системи наведена на рисунку 3.1.

З неї ми бачимо, що існують дві сторони процесу автентифікації. З однієї сторони це ЕОМ з операційною системою, у якій встановлений драйвер USB-ключа. З іншої сторони це користувач з USB-ключем, який потребує процесу автентифікації для доступу до системи.

На стороні ПЕОМ, крім драйвера USB-ключа, існує:

– Блок генерування частини одноразового паролю.

– Блок автоматичної генерації паролю.

– Блок перевірки ПІН-кода доступу до USB-ключа користувача.

– Блок синхронізації по часу.

– Блок підрахунку кількості повторів.

– Блок журналювання.

– Блок вибору методу автентифікації.

– БД користувачів з визначенням їх прав доступу.

На стороні флеш-накопичувача існує:

– Блок криптографічних перетворень.

– БД параметрів користувача.

Процедура автентифікації відбувається наступним чином:

1. На ПЕОМ встановлюється драйвер USB-ключа.

2. Користувач під’єднує при вході у систему USB-ключ, для початку процедури автентифікації.

3. Система видає запит ПІН-коду доступу до USB-ключа.

4. Після введення ПІН-коду, видається вікно у якому потрібно ввести логін та пароль, необхідний для виконання процедури автентифікації й допуску користувача до системи.

5. Після введення паролю, на його основі формується у блоці генератора ключів пароль, частина якого відсилається до флеш-накопичувача. При записі профілю користувача до пам'яті USB-ключа пароль може генеруватися автоматично або вводитися вручну. При автоматичній генерації формується випадковий, довжиною до 128 символів, пароль. При цьому користувач не буде знати свій пароль і не зможе ввійти в систему без USB-ключа. Вимога використання тільки автоматично згенерованих паролів може бути настроєна як обов'язкова.

6. Програмне забезпечення, встановлене на флеш-накопичувачі, згідно заданих таємних криптографічних алгоритмів перетворює цю частину ключа й надсилає відповідь.

7. На ПЕОМ відбуваються аналогічні перетворення, які заносяться у зашифрованому вигляді, до БД користувачів, та паролів.

8. Отримані з флеш-накопичувача дані порівнюються, з тими, які записані у БД.

9. Якщо дані співпадають, то користувач отримує доступ до системи з наданими йому правами. У іншому випадку, надається відмова у доступі.

Кількість повторів обмежується трьома спробами.

Усі дії заносяться до журналу подій (у лог-файл). У випадку підозрілих дій видається сигнал адміністраторові ПЕОМ.

Рисунок 3.3 Структурна схема системи автентифікації з використанням USB-ключа

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.4.

З рисунку видно, що розроблена система складається з наступних частин, які реалізують типи прав доступу до USB-ключа:

1. Гостьовий – надає наступні можливості:

– можливість переглядати об'єкти у відкритій області пам'яті;

– можливість одержання із системної області пам'яті загальної інформації відносно USB-ключа, що включає ім'я USB-ключа, ідентифікатори й деякі інші параметри.

При гостьовому доступі знання PIN-коду не обов'язково.

2. Користувальницький – надає наступні можливості:

– право переглядати, змінювати й видаляти об'єкти в закритих, відкритих і вільній областях пам'яті;

– можливість одержання загальної інформації відносно USB-ключа;

– право міняти PIN-код і перейменовувати USB-ключ;

– право налаштовувати параметри кешування змісту закритої області пам'яті й додаткового захисту закритих ключів паролем (при відсутності пароля адміністратора або з дозволу адміністратора)

– право перегляду й видалення сертифікатів у сховищі USB-ключа і ключових контейнерах RSA.

3. Адміністраторський – надає наступні можливості:

– право міняти PIN-код користувача, не знаючи його;

– право зміни пароля адміністратора;

– право налаштовувати параметри кешування змісту закритої області пам'яті й додаткового захисту закритих ключів паролем, а також можливість робити ці настроювання доступними в користувальницькому режимі.

4. Ініціалізаційний – право форматувати USB-ключ.

Рисунок 3.4 – Функціональна схема системи

Для одержання доступу до даних, що зберігається в пам'яті USB-ключа, необхідно ввести PIN-код (Personal Identification Number). В PIN-коді не рекомендується використовувати пробіли й кирилицю. При цьому PIN-код повинен задовольняти критеріям якості, заданим у файлі %systemroot%\system32\etcpass.ini.

Редагування цього файлу, що містить критерії якості PIN-коду, здійснюється за допомогою утиліти USB-ключа.

Адміністраторський доступ до USB-ключа може бути зроблений тільки після правильного введення пароля адміністратора. Якщо ж у процесі форматування пароль адміністратора не заданий, то звернутися із правами адміністратора не можна.

За допомогою розробленого програмного забезпечення можна:

– налаштовувати параметри USB-ключа і його драйверів;

– переглядати загальну інформацію відносно USB-ключа;

– імпортувати, переглядати й видаляти сертифікати (за винятком сертифікатів зі сховища USB-ключів) і ключові контейнери RSA;

– форматувати USB-ключ;

– налаштовувати критерії якості PIN-кодів.

Для установки даного програмного забезпечення необхідні права локального адміністратора. Варто пам'ятати, що до установки розробленого програмного забезпечення не можна підключати USB-ключ.

Якщо на комп'ютері встановлене програмне забезпечення, підключите USB-ключ до порту USB або до подовжувального кабелю. Після цього почнеться процес обробки нового обладнання, що може зайняти якийсь час. По завершенні процесу обробки нового обладнання на USB-ключ засвітиться світловий індикатор.

Утиліта "Властивості USB-ключа" дозволяє виконувати основні операції по керуванню токенами, такі як зміна паролів, перегляд інформації й сертифікатів, розташованих у пам'яті USB-ключа. Крім того, за допомогою

утиліти "Властивості USB-ключа" можна швидко й легко переносити сертифікати між комп'ютером і USB-ключем, а також імпортувати ключі до пам’яті USB-ключа.

Кнопка "Розблокувати" необхідна, якщо користувач забув свій PIN-код, і не може прийти до адміністратора USB-ключа (наприклад, користувач перебуває у відрядженні). Звернувшись до адміністратора по e-mail, користувач зможе одержати для цього USB-ключа від адміністратора шестнадцятирічний запит, сформований на підставі даних, що зберігаються в базі TMS, занесши який у поле "відповідь" користувач одержить доступ на зміну PIN-коду.

При зміні PIN-коду необхідно щоб новий PIN-код відповідав вимогам якості введеного пароля. Якість пароля перевіряється відповідно до введених критеріїв.

Для того щоб перевірити відповідність пароля обраним критеріям, введіть пароль у рядок. Під цим рядком виводиться інформація про причини невідповідності введеного пароля обраним критеріям у відсотках, а також графічно й у відсотках умовно відображається якість введеного пароля відповідно до обраних критеріїв.

У таблиці наведені критерії якості PIN-коду й зазначені їхні значення, що налаштовуються. Як значення критерію може бути використана негативна величина, виражена у відсотках. Таке значення називається штрафом.

Таблиця 3.2 – Критерії якості PIN-коду

Стандарт	Опис	Можливі значення	Значення за замовчуванням
ABCOrder	PIN-код містить послідовність символів за абеткою	-100% – 0%	-10%
ABCOrderBase	Довжина послідовності символів для критерію ABCOrder	2% – 100%	3%

Продовження таблиці 3.2

Стандарт	Опис	Можливі значення	Значення за замовчуванням
CheckCurrPass	Новий пароль дорівнює поточному	-100% – 0%	-100%
CheckDictionary	Пароль зі словника	-100% – 0%	-100%
CheckOldPasses	Новий пароль дорівнює одному з попередніх	-100% – 0%	0%
DefaultPassChange	Зміна пароля, прийнятого за замовчуванням	None (зміна пароля не потрібна) Warning (виводиться повідомлення із попередженням) Enforce (використання пароля за замовчуванням неможливо)	Enforce
Dictionary	Файл словника	Абсолютний шлях до файлу словника	Не задано
DigitsOnly	Пароль тільки із цифр	-100% – 0%	-5%
Duplicates	Наявність двох однакових символів	-100% – 0%	-20%

Продовження таблиці 3.2

Стандарт	Опис	Можливі значення	Значення за замовчуванням
Expiry	Термін дії до появи попередження про зміну пароля	0 – 3650 днів	360
ExpiryEnforce	Максимальний термін дії в днях	0 – 3650	0 (не встановлений)
KeyboardProximity	Наявність декількох символів у тім же порядку, як на клавіатурі	-100% – 0%	-10%
KeyboardProximity Base	Довжина послідовності символів для попереднього параметра	2 – 100	3
LikeDictionary	Пароль схожий на пароль зі словника	-100% – 0%	-80%
MinChangePeriod	Мінімальний термін дії в днях	0 – 3650	0 (не встановлений)
MinimalLength	Мінімальна довжина в символах	0 – 100	4
MinimalQuality	Мінімальна стійкість у відсотках	0 – 100	30
NoDigits	Відсутність цифр	-100% – 0%	-5%
NoLowerCase	Відсутність малих літер	-100% – 0%	-5%

Продовження таблиці 3.2

Стандарт	Опис	Можливі значення	Значення за замовчуванням
NonPrintable	Використання букв українського алфавіту, що не друкуються й деяких службових символів	-100% – 0%	-100%
NoPunctuation	Відсутність розділових знаків і службових символів	-100% – 0%	-5%
NoUpperCase	Відсутність прописних букв	-100% – 0%	-5%
OptimalLength	Довжина, що рекомендується, у символах	0 – 100	12
PhonesandSerialNumbers	Використання в паролі номерів телефонів, серійних номерів і т.п.	-100% – 0%	-5%
Repeating	Наявність повторюваних символів	-100% – 0%	-20%
SaveOldPasses	Кількість використаних раніше паролів, що зберігаються в пам'яті USB-ключа для перевірки за критерієм CheckOldPasses	0-20	3
SmallPassword	Довжина пароля менше WarningLength	-100% – 0%	-5%

Продовження таблиці 3.2

Стандарт	Опис	Можливі значення	Значення за замовчуванням
WarningLength	Якщо довжина пароля менше цього параметра, при перевірці якості пароля видається попередження	0-100	6
WhiteSpaces	Пароль містить символи пробілу	-100% – 0%	-100%

Для того щоб задати список неприпустимих або небажаних паролів, створіть текстовий файл. Або можливо скористатися так званими частотними словниками, які використовуються для підбора паролів. Файли таких словників можна взяти на сайті www.passwords.ru.

Приклад такого словника:

– anna

– annette

– bill

– password

– william

Призначте критерію "Dictionary" шлях до створеного файлу. При цьому шлях до файлу словника на кожному комп'ютері повинен збігатися зі значенням критерію "Dictionary".

Вхід у систему за допомогою USB-ключа

При автентифікації в Windows використовуються ім'я користувача й пароль, що зберігаються в пам'яті USB-ключ. Це дає можливість застосовувати строгу автентифікацію на основі токенів.

Разом з тим хотілося б додати, що у великих компаніях, що використовують доменну структуру, необхідно подумати про впровадження PKI і централізованому застосуванні SmartCardLogon.

При використанні USB-ключа можуть застосовуватися нікому не відомі випадкові складні паролі. Крім того, передбачена можливість використання сертифікатів, що зберігаються в пам'яті USB-ключа, для реєстрації на основі смарт-карт, що підвищує безпека входу в Windows.

Це стало можливим завдяки тому, що система Windows 10/11 дозволяє використовувати різні механізми доступу, що заміняють метод автентифікації за замовчуванням. Механізми ідентифікації й автентифікації служби входу в Windows (winlogon), що забезпечує інтерактивну реєстрацію в системі, убудовані в замінну бібліотеку, що приєднується динамічно (DLL), іменовану GINA (Graphical Identification and Authentication, робочий стіл автентифікації). Коли система має потребу в іншому методі автентифікації, який би замінив механізм "ім'я користувача/пароль" (використовуваний за замовчуванням) стандартну msgina.dll заміняють новою бібліотекою.

При установці USB-ключа заміняється бібліотека робочого стола автентифікації й створюються нові параметри реєстру. GINA відповідає за політику інтерактивного підключення й здійснює ідентифікацію й діалог з користувачем. Заміна бібліотеки робочого стола автентифікації робить USB-ключ основним механізмом перевірки дійсності, що розширює можливості стандартної автентифікації Windows 10/11, заснованої на застосуванні ім'я користувача й пароля.

Користувачі можуть самостійно записувати до пам’яті USB-ключ інформацію, необхідну для входу в Windows (профілі), якщо це дозволено політикою безпеки підприємства. Профілі можна створювати за допомогою майстра створення профілів USB-ключа Windows Logon.

USB-ключ SecurLogon автентифікує користувача Windows 10/11 с допомогою USB-ключ, використовуючи або сертифікат користувача зі смарт-

картою, або ім'я користувача й пароль, які зберігаються в пам'яті USB-ключа. USB-ключ містить у собі всі необхідні файли й драйвери, що забезпечують підтримку USB-ключа в Windows Logon.

Всі нові USB-ключ мають той самий PIN-код, установлений за замовчуванням при виробництві. Цей PIN-код 1234567890. Для забезпечення строгої, двофакторної автентифікації й повної функціональності користувач обов'язково повинен замінити PIN-код за замовчуванням власним PIN-кодом відразу після одержання нового USB-ключ.

Важливо: PIN-код не слід плутати з паролем користувача Windows.

Установка

Для того щоб установити USB-ключ Windows Logon:

– увійдіть у систему як користувач із правами адміністратора;

– двічі клацніть SecurLogon.msi;

– з'явиться вікно майстра установки USB-ключ SecurLogon;

– натисніть кнопку "Next", з'явиться ліцензійна угода USB-ключ Enterprise;

– прочитайте угоду, натисніть кнопку "I accept" (Приймаю), а потім кнопку "Next";

– наприкінці установки виробляється перезавантаження.

Використання USB-ключ SecurLogon

USB-ключ SecurLogon дозволяє користувачам реєструватися в Windows 10/11/2003 за допомогою USB-ключа із записаним у пам'яті паролем.

Зміна пароля

Можливо перемінити пароль Windows після входу в систему за допомогою USB-ключ. Для того щоб перемінити пароль після входу в систему за допомогою USB-ключ:

– увійдіть у систему, використовуючи USB-ключ;

– натисніть "CTRL+ALT+DEL", з'явиться вікно "Безпека Windows / Windows Security";

– Клацніть кнопку "Зміна пароля / Change Password", якщо поточний пароль був створений вручну, те з'явиться вікно "Зміна пароля / Change Password", але якщо поточний пароль був створений випадковим образом, то переходимо до пункту 5;

– Уведіть новий пароль у полях "Новий пароль / New Password" і "Підтвердження / Confirm New Password" і натисніть кнопку "OK";

– Якщо поточний пароль був створений випадковим образом, то й новому паролі буде створений автоматично;

– у діалоговому вікні, що з'явилося, введіть PIN-код USB-ключ і натисніть кнопку "OK"

– з'явиться вікно з підтверджувальним повідомленням.

Блокування робочої станції

Можливо забезпечувати безпеку вашого комп'ютера, не виходячи із системи, шляхом блокування комп'ютера. При від'єднанні USB-ключа від порту USB або кабелю (після вдалої реєстрації) операційна система автоматично заблокує ваш комп'ютер.

Для того щоб розблокувати ваш комп'ютер:

Коли ваш комп'ютер заблокований, з'являється вікно "Блокування комп'ютера Computer Locked". Підключите USB-ключ до порту USB або кабелю. У вікні, що з'явилося, введіть PIN-код у поле " USB-ключ Password" і натисніть кнопку "OK" – комп'ютер розблокований. У випадку натискання "CTRL+ALT+DEL" і введення пароля комп'ютер буде розблокований без використання USB-ключа.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.5.

З неї ми бачимо, що у системі взаємодіють наступні процеси.

Спершу завантажується процес початку/кінця роботи програми.

Він взаємодіє з наступними процесами:

– Процес вибору методу автентифікації.

– Процес перегляду та видалення сертифікатів у сховищі USB-ключа.

Процес вибору методу автентифікації взаємодіє з наступними процесами:

– Процес встановлення PIN-коду користувача.

– Процес встановлення пароля адміністратора.

– Процес налаштування параметрів кешування.

Останній процес у свою чергу взаємодіє з процесом налаштування додаткового захисту закритих ключів паролем.

Процес налаштування додаткового захисту закритих ключів паролем взаємодіє з процесом вибору прав доступу.

Рисунок 3.5 – Діаграма процесів розробленої системи

Процес вибору прав доступу реалізує після процедури автентифікації, надання тих або інших прав доступу, для реалізації яких запускаються наступні процеси:

– Процес гостьового доступу.

– Процес користувальницького доступу.

Процес гостьового доступу взаємодіє з наступними двома процесами:

– Процесом форматування USB-ключа.

– Процесом перегляду об’єктів у відкритій області пам’яті.

У свою чергу процес надання користувальницького доступу взаємодіє з процесом перегляду та зміни об’єктів у закритих та відкритих областях пам’яті.

Процес перегляду та зміни об’єктів у закритих та відкритих областях пам’яті взаємодіє з процесом зміни PIN-коду та перейменування USB-ключа.

Процес зміни PIN-коду та перейменування USB-ключа взаємодіє з процесом перегляду та видалення сертифікатів у сховищі USB-ключа.

Цей процес є завершальним у системі.

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. Його блок-схема зображена на рисунку 4.1.

З рисунку видно, що після запуску програми спочатку відбувається вивід основного вікна програми.

Потім здійснюється вибір користувачем методу автентифікації.

За цією дією слідує налаштування USB-ключа.

Після налаштування USB-ключа реалізовано налаштування параметрів кешування.

Наступною дією є налаштування додаткового захисту закритих ключів паролем.

Після цього система визначає під якими правами реалізовано вхід у систему.

Якщо вхід реалізовано під правами користувача, то відбуваються наступні дії.

Завантажується блок перевірки ПІН-коду доступу до USB-ключа.

Відбувається перевірка дійсності ПІН-коду.

Якщо він недійсний, то виводиться повідомлення про те, що введений ПІН-код не є дійсним.

У іншому випадку, відбуваються наступні дії:

– Перегляд та зміна об’єктів у відкритій та закритій областях пам’яті.

– Управління сертифікатами та USB-ключем.

На цій операції блок реалізації прав користувача завершується.

Користувач має право на реалізацію гостьового доступу.

У цьому випадку відбувається перегляд об’єктів у відкритій області пам’яті.

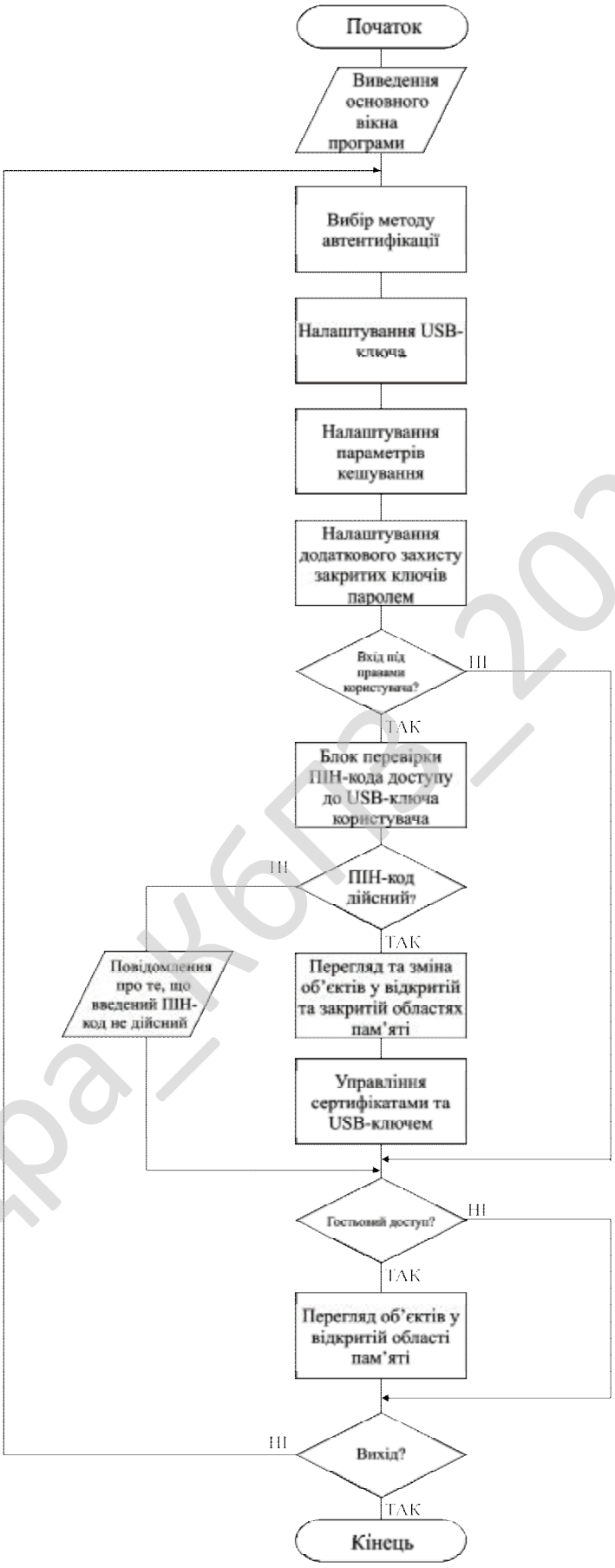

Рисунок 4.1 – Блок-схема основної програми

Після виконання усіх перерахованих вище дій користувач обирає працювати йому далі з програмним об’єктом, або завершити роботу програмного забезпечення.

Реалізація центру розподілу ключів відбувається за допомогою технології інфраструктури відкритих ключів, та з використанням алгоритму RSA.

Алгоритм працює наступним чином.

Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток N = P*Q і значення функції φ(N) = (P-1)(Q-1). Далі відправник обчислює число E з умов E<φ (N), НЗД (E, φ(N))=1 і число D з умов D<N, E*D≡1(mod ((N)).

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою хеш-функції h(-) у ціле число m: m = h(M).

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи хеш-значення m і секретний ключ D: S = m (mod N).

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює хеш-значення повідомлення M двома різними способами. Насамперед, він відновлює хеш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E \pmod N$.

Крім того, він знаходить результат хешування прийнятого повідомлення M с допомогою такої ж хеш-функції h(-): m = h(M).

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то

одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти модуль N на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D, що відповідає відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" що підписав.

На рисунку 4.2 зображена блок-схема підпрограми шифрування за допомогою алгоритму RSA.

Розглянемо, як вона працює.

Спершу відбувається формування у центрі сертифікації (ЦС) ключів та параметрів з урахуванням модифікацій.

Після цього відбувається видача центру сертифікації таємного ключа D_k, параметра N = P*Q, ключа симетричного шифру.

Якщо ці параметрине відповідають критеріям центру сертифікації, то відбувається повторна видача цих параметрів.

У протилежному випадку, тобто, якщо парамети відповідають вимогам центру сертифікації, то відбувається шифрування інформації відкритим шифром.

Після цього відбувається формування відкритої части ни цифрового підпису (ЦП):

– Хеш-функція.

– Ключ.

– Час.

– Дані відправника.

Наступною дією є накладання цифрового підпису, за наступним правилом ЕЦП = ВЕЦП ^{Dk} (mod N), де ВЕЦП – відкритий електронний цифровий підпис.

Після цього відбувається запис зашифрованих даних, та електронного цифрового підпису. На цьому підпрограма закінчує свою роботу.

Рисунок 4.2 – Блок-схема підпрограми шифрування за допомогою алгоритму RSA

На рисунку 4.3 зображена блок-схема підпрограми дешифрування за допомогою алгоритму RSA.

Вона працює наступним чином.

Спершу відбувається отримання даних з каналу зв'язку, під яким у даному випадку підрозумівається отримання логіна та пароля користувача. З метою визначення його повноважень.

Після цього відбувається видача центром сертифікації наступних параметрів:

– Відкритого ключа E_k.

– Параметра N = P*Q, де P та Q – великі прості числа.

– Ключа шифрування симетричного шифру.

Наступним кроком після отримання цих параметрів є дешифрування цифрового підпису за допомогою наступного вираження:

$$ВЕЦП = ЕЦП^{Ek}(\ mod\ N),\ де\ E_k – відкритий\ ключ\ ЦП.$$

При цьому потрібно пам'ятати, що відкритий та закритий ключ користувача у центрі сертифікації взаємодіють наступним чином: $E_k*D_k = 1(mod\ \varphi\ (N\ ))$, де $\varphi\ (N\ ) = (P-1)*(Q-1)$.

Наступним кроком, є перевірка коректності цифрового підпису.

Якщо він некоректний, то відбувається запит до передавача на повтор пересилки даних, та усі вищеперераховані кроки повторюються ще раз.

Якщо ж він коректний, то відбувається дешифрування зашифрованої інформації.

В зв'язку з тим, що стійкість алгоритму RSA базується на неможливості розкладення на множники модуля N (факторизації), в данній магістерській роботі пропонується модифікувати алгоритм наступним чином. Замість великих простих P та Q вибирати для центра сертифікації відкритого та закритого ключів алгоритма RSA сильні великі прості числа P_i та Q_i тобто такі числа, які володіють наступними властивостями: $P_i + 1 = L*M$, $P_i – 1 = X*Y$, $Q_i + 1 = W*R$, $Q_i + 1 = F*G$, де L, M, X, Y, W, R, F, G – великі прості числа.

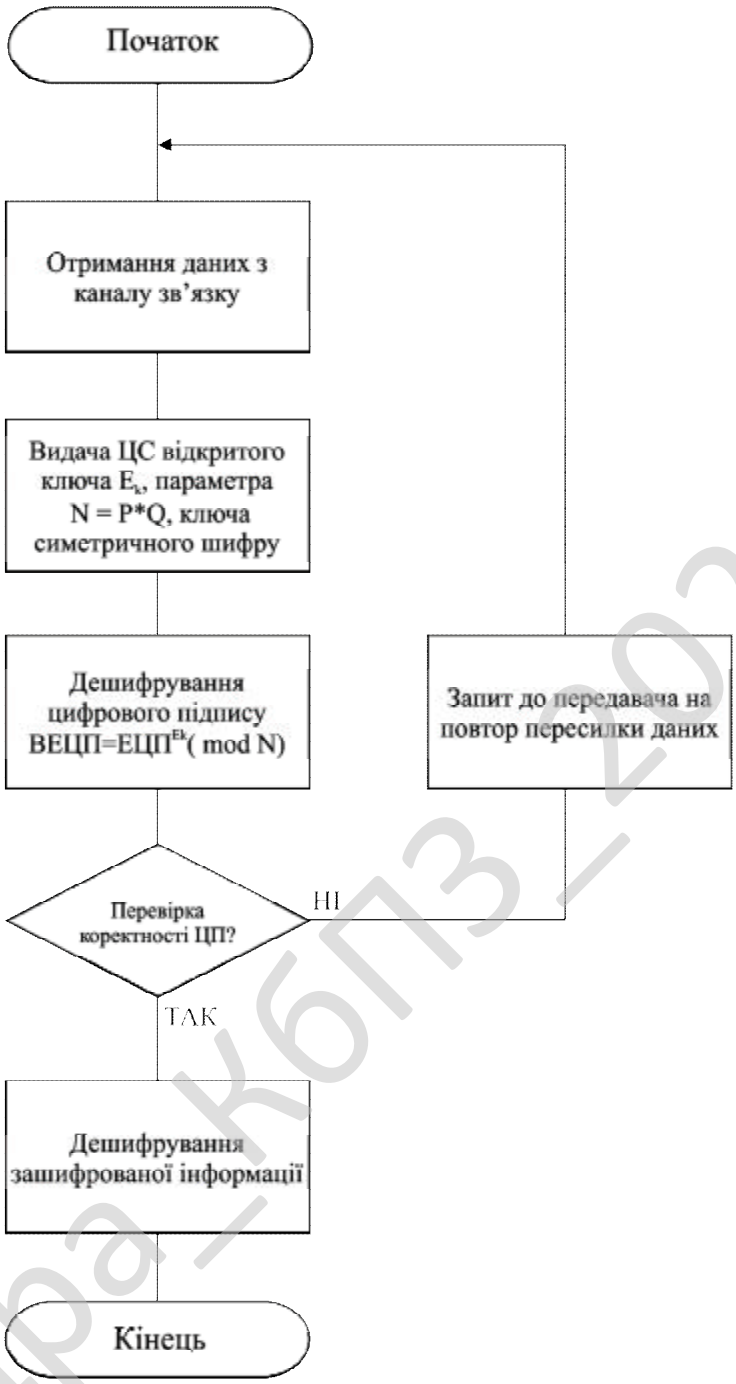

Рисунок 4.3 – Блок-схема підпрограми дешифрування за допомогою алгоритму RSA

Розгортання служб сертифікації

Для впровадження системи сертифікації й керування відкритими ключами в системі автентифікації користувачів пропонується використовувати дворівневу ієрархію центрів сертифікації: ізольований ЦС (Stand-alone Root CA) як центр

сертифікації й ізольований підлеглий ЦС (Stand-alone subordinate CA) як центр реєстрації.

Розгортання системи сертифікації й керування відкритими ключами виконується в наступній послідовності:

– установка центра сертифікації;

– конфігурування центра сертифікації;

– установка й настроювання центра реєстрації;

– установка й настроювання сховища сертифікатів;

– установка й настроювання допоміжних систем і додатків.

Центр сертифікації поєднує людей, процеси, програмні й апаратні засоби, залучені в безпечне зв'язування імен користувачів і їхніх відкритих ключів. Центр сертифікації відомий суб'єктам інфраструктура відкритих ключів алгоритму RSA по двох атрибутах: назві й відкритому ключу. Центр сертифікації включає своє ім'я в кожний випущений їм сертифікат і в **список анульованих сертифікатів** (CAC) і підписує їх за допомогою власного секретного ключа. Користувачі можуть легко ідентифікувати сертифікати по ім'ю центр сертифікації і переконатися в їхній дійсності, використовуючи його відкритий ключ. Центр сертифікації – головний керуючий компонент інфраструктури відкритих ключів алгоритму RSA – виконує наступні основні функції:

– формує власний секретний ключ; якщо є головним центр сертифікації, то видає й підписує свій сертифікат, який називається **самовиданим** або **самопідписаним**;

– випускає (тобто створює й підписує) сертифікати відкритих ключів підлеглих центрів, що засвідчують, і кінцевих суб'єктів інфраструктури відкритих ключів алгоритму RSA; може випускати крос-сертифікати, якщо зв'язано відносинами довіри з іншими інфраструктурами відкритих ключів алгоритму RSA;

– підтримує реєстр сертифікатів (базу всіх виданих сертифікатів) і формує списки CAC із регулярністю, певної регламентом сертифікаційного центра;

– публікує інформацію про статус сертифікатів і списків CAC.

Реєстраційний центр (РЦ) є необов'язковим компонентом інфраструктури відкритих ключів алгоритму RSA. Звичайно РЦ одержує від центра, що засвідчує, повноваження реєструвати користувачів, забезпечувати їхня взаємодія із центр сертифікації і перевіряти інформацію, що заноситься в сертифікат. Сертифікат може містити інформацію, що надана суб'єктом, що подає заявку на сертифікат і пред'являє документ (паспорт, права водія, чекову книжку й т.п.) або третьою стороною (наприклад, кредитним агентством – про кредитний ліміт пластикової карти). Іноді в сертифікат включається інформація з відділу кадрів або дані, що характеризують повноваження суб'єкта в компанії (наприклад, право підпису документів певної категорії). РЦ агрегує цю інформацію й надає її центр сертифікації.

Репозиторій – спеціальний об'єкт інфраструктури відкритих ключів, база даних, у якій зберігається реєстр сертифікатів. Репозиторій значно спрощує керування системою й доступ до ресурсів. Він надає інформацію про статус сертифікатів, забезпечує зберігання й поширення сертифікатів і CAC, управляє внесеннями змін у сертифікати. До репозиторію пред'являються наступні вимоги:

– простота й стандартність доступу;

– регулярність відновлення інформації;

– вбудована захищеність;

– простота керування;

– сумісність із іншими сховищами (необов'язкова вимога).

Архів сертифікатів виконує функцію довгострокового зберігання (від імені сертифікаційного центра) і захисту інформації про всі видані сертифікати. Архів підтримує базу даних, використовувану при виникненні споровши із приводу надійності електронних цифрових підписів, якими в минулому засвідчувалися документи. Архів підтверджує якість інформації в момент її одержання й забезпечує цілісність даних під час зберігання. Інформація, надавана центр сертифікації архіву, повинна бути достатньої для визначення статусу

сертифікатів і їхнього видавця. Архів повинен бути захищений відповідними технічними засобами й процедурами.

Кінцеві суб'єкти, або користувачі, інфраструктури відкритих ключів алгоритму RSA діляться на дві категорії: власники сертифікатів і сторони, що довіряють. Вони використовують деякі сервіси й функції, щоб одержати сертифікати або перевірити сертифікати інших суб'єктів. Власником сертифіката може бути фізична або юридична особа, додаток, сервер і т.д. сторони, Що Довіряють, запитують і покладаються на інформацію про статус сертифікатів і відкритих ключів підпису своїх партнерів по діловому спілкуванню.

Операційні протоколи – це протоколи для доставки сертифікатів (або інформації про їхній статус) і списків анульованих сертифікатів до клієнтських систем, що використовують сертифікати.

Протоколи керування необхідні для підтримки взаємодій між користувачем інфраструктури відкритих ключів і суб'єктами керування. Протоколи керування підтримують:

– реєстрацію суб'єкта для одержання сертифіката;

– ініціалізацію (наприклад, генерації пари ключів);

– випуск сертифіката;

– відновлення пари ключів;

– відновлення пари ключів після закінчення терміну дії сертифіката;

– обіг із запитом про анулювання сертифіката;

– крос-сертифікацію, коли два сертифікаційних центри обмінюються інформацією для генерації крос-сертифіката.

Політика застосування сертифікатів і регламент центра сертифікації є в документах, де приведенні зобов'язання сторін і правила використання сертифікатів.

Загальна схема функціонування інфраструктури відкритих ключів алгоритму RSA працює наступним чином. Користувач відправляє запит на сертифікат у РЦ (транзакція керування). Якщо запит фактично схвалений, то

направляється безпосередньо в центр сертифікації для завірення цифровим підписом. Центр сертифікації перевіряє запит на сертифікат, і якщо той проходить верифікацію, то підписується й випускається сертифікат. Сертифікат публікується в репозиторії; залежно від конкретної конфігурації інфраструктури відкритих ключів алгоритму RSA, ця функція може бути покладена на реєстраційний або центр, що засвідчує. Процес анулювання сертифіката аналогічний процесу його генерації. Кінцевий суб'єкт запитує центр сертифікації про анулювання свого сертифіката, РЦ приймає рішення й направляє запит про анулювання в центр сертифікації. Центр сертифікації вносить зміни в список анульованих сертифікатів і публікує його в репозиторії. Кінцеві суб'єкти можуть перевірити статус конкретного сертифіката через операційний протокол

Поточні завдання системи сертифікації й керування відкритими ключами

До завдань, постійно виконуваних системою сертифікації й керування відкритими ключами, відносяться:

- архівування й відновлення центрів сертифікації;
- відмова або схвалення запитів на сертифікати;
- відкликання сертифікатів;
- публікація списків відкликаних сертифікатів (CRL);
- відновлення сертифікатів центрів сертифікації;
- відновлення сертифікатів користувачів.

Відновлення після збою

Серйозні збої, такі, як відмова жорсткого диска або компрометація сертифіката ЦС, здатні повністю порушити роботу служб сертифікації. Існує кілька шляхів мінімізації негативних наслідків таких збоїв і забезпечення своєчасного відновлення. До операцій по зниженню ризику відмови або компрометації ЦС відносяться:

- захист закритих ключів центрів сертифікації;
- розробка планів відновлення.

Наступні операції дозволяють попередити ризик відмови ЦС і звести до мінімуму час простою служб ЦС:

– створення дублюючих ЦС;

– часте архівування ЦС, що дозволяє швидко й з мінімальними втратами даних відновити центр сертифікації;

– установка служб сертифікації на дискових масивах і масивах RAID-5;

– розробка планів відновлення й навчання адміністраторів виконанню цих планів;

– збереження дані конфігурації ЦС для безпроблемного й точного відновлення конфігурації у випадку збою.

Забезпечення безпеки центрів сертифікації

Комп'ютери, на яких працюють ЦС, – це найбільш імовірні мети для атаки зловмисників, що намагаються порушити роботу служб або скомпрометувати захист і інформаційних систем. Одержавши доступ до ЦС або скориставшись недоліками захисту, ті що атакують можуть одержати доступ до ресурсів і скомпрометувати безпеку цілого ланцюжка довіри. Тому ЦС мають потребу в більше надійному захисті, чим звичайні.

Ризик атаки на ЦС залежить від багатьох факторів, у тому числі від захищеності , цілей атаки й витрат на неї. До можливих негативних наслідків компрометації ЦС відносяться:

– втрата інформації, що становить інтелектуальну власність організації;

– відмова або простій служб;

– ушкодження або руйнування ресурсів;

– витрати на усунення наслідків компрометації ЦС і повторне розгортання центрів сертифікації й сертифікатів.

Компрометація кореневого ЦС обійдеться набагато дорожче, ніж проміжного або випускаючого ЦС. Щоб знизити потенційні збитки, варто створити в організації ієрархію з декількох ЦС.

Визначаючи адекватність мер безпеки, необхідно зважити й оцінити

витрати на ці заходи щодо забезпечення безпеки й можливі збитки у випадку компрометації ЦС. До заходів щодо забезпечення безпеки ЦС відносяться:

– розміщення у захищених центрах даних і надання фізичного доступу до них тільки довіреним адміністраторам;

– використання апаратних центрів сертифікації або апаратних постачальників CSP для забезпечення максимального захисту закритих ключів ЦС;

– конфігурування параметрів безпеки для забезпечення високого рівня захисту, такого, як рівень High Security (Високий) захисту шаблонів;

– використання службової програми Windows 2000 System Key (SysKey) для забезпечення шифрування захищених сховищ ЦС;

– аудит безпеки для контролю й спостереження за можливими атаками на ЦС;

– обмеження надання прав користувачам шляхом надання прав тільки відповідній групі адміністраторів (іншим користувачам або групам треба заборонити перегляд або виконання будь-яких завдань на локальному комп'ютері зі ЦС);

– відключення непотрібних служб на х зі ЦС; працюючі служби являють собою додаткові «лазівки» для зловмисників;

– розгортання політики й виконання процедур безпеки при розгортанні ЦС на підприємстві.

При виборі мір безпеки ЦС варто зважити витрати на їхню реалізацію й підтримку, з одного боку, і ризик нападу на ЦС і можливі збитки від компрометації ЦС, з іншої сторони. У загальному випадку чим вище ризик нападу й збитки від компрометації, тим більше витрати на заходи щодо забезпечення безпеки ЦС. Максимальний захист варто забезпечити кореневим ЦС, а проміжні ЦС треба убезпечити надійніше, ніж випускаючі ЦС.

Допустимо, що в організації ухвалено рішення захистити великий обсяг надзвичайно коштовної й конфіденційної інформації, використавши рішення на

основі відкритих ключів. Також вирішено придбати дорогий апаратний ЦС для виконання функцій кореневого ЦС і розмістити його для безпеки в захищеному сховищі, розташованому в головному офісі організації. Доступ до кореневого ЦС, що сертифікує всі проміжні ЦС у підрозділах, надається тільки довіреним адміністраторам. Проміжні ЦС – це ізольовані ЦС на комп'ютерах під керуванням Windows XP, від'єднаних від й розміщених у захищених центрах даних під наглядом адміністратора підрозділу. Проміжний ЦС використовується в міру необхідності для сертифікації випускаючих ЦС під керуванням Windows XP відповідно до потреб кожного підрозділу. Випускаючі ЦС – це ЦС Windows XP підприємства або ізольовані ЦС, розміщені в захищених центрах даних кожного підрозділу. Політика безпеки організації повинна мати на увазі самі строгі міри безпеки виконання запиту, авторизації й впровадження кореневого, проміжних і випускаючих ЦС на підприємстві й контроль за ними.

З іншого боку, якщо в організації рішення безпеки на базі відкритого ключа застосовуються для захисту не дуже коштовної інформації, цілком достатньо ізольованого кореневого ЦС Windows XP, розміщеного в центрі даних, а не згаданого в попередньому прикладі дорогого апаратного ЦС, розміщеного в захищеному сховищі. При цьому припустимо розмістити проміжні й випускаючі ЦС у підрозділах за межами центра даних. Також не потрібно таких строгих обмежень на впровадження, запити й авторизацію ЦС.

Служби сертифікації Windows XP на базі постачальника Microsoft Base CSP зможуть задовольнити більшість потреб у захисті ЦС. Для забезпечення найвищої безпеки ЦС варто використовувати апаратні ЦС.

Захист закритих ключів алгоритму RSA центрів сертифікації

Якщо в зловмисника є доступ до комп'ютера ЦС – безпосередньо або по , він у стані розшифрувати особистий ключ і потім виступати від імені ЦС і одержувати доступ до коштовних ресурсів. Він зможе красти інформацію, порушити роботу служб і знищити ресурси. Скомпрометований ключ алгоритму RSA ЦС підриває й зводить «на ні» всю систему безпеки, забезпечену

цим ЦС, і всю його ієрархію ЦС. Необхідно регулярно проводити заходу щодо зниження атаки на ключі ЦС.

Необхідно забезпечити захист із центрами сертифікації, як описано раніше в цьому розділі. Забезпечення фізичної безпеки мінімізує ризик одержання атакуючого доступу до ЦС або захищеному сховищу (будь воно апаратним або програмним), де зберігається ключ алгоритму RSA ЦС. Забезпечення безпеки й (програмного забезпечення) знижує ризик того, що порушники одержать доступ до ЦС або скористаються недоліками додатків або служб цього для компрометації ключа ЦС. Необхідно забезпечити посилений захист ключів центра сертифікації. Якщо потрібна максимальна безпека закритих ключів, необхідно скористатися апаратними постачальниками CSP, тому що в цьому випадку ключі зберігаються на стійкі до злому апаратних пристроях і ніколи не надаються операційній системі. Необхідно використовувати службову програму SysKey для забезпечення додаткового захисту закритих ключів ЦС, збережених постачальниками Microsoft CSP.

Використання в центрах сертифікації ключів великої довжини знижує ризик атаки на ключ алгоритму RSA, однак довгі ключі вимагають більше дискового простору й обчислювальних потужностей для підписання сертифікатів. Варто вибрати максимальну можливу довжину ключа й урахувати обмеження на пам'ять і продуктивність ЦС.

Наприклад, 4096-бітний ключ алгоритму RSA ЦС забезпечує чудову безпеку, але підписання сертифікатів таким довгим ключем займає занадто багато часу навіть при наявності плати криптоакселератора. Такий ключ алгоритму RSA цілком придатний для коренев або проміжного ЦС, які використовуються нечасто й тільки для сертифікації підлеглого ЦС. Для більшості випускаючих ЦС 4096-бітних ключів неприйнятний через неприпустиме зниження продуктивності роботи ЦС. На випускаючих ЦС варто використовувати ключі, які забезпечують задовільну безпеку, не сповільнюють роботу ЦС і відповідають довгостроковим цілям конкретної служби сертифікації. Для підвищення продуктивності

випускаючого ЦС і використання більше довгих ключів рекомендується встановити плату криптоакселератора. Перш ніж розгортати ЦС на підприємстві, необхідно протестувати його продуктивність для різних довжин ключів у лабораторних умовах і на пілотних системах.

Необхідно визначити адекватні терміни дії ключів ЦС. Чим більше термін дії ключа, тим вище ризик його компрометації, тому що в атакуючих більше часу на злом. Не існує простого правила визначення максимальних термінів дії ключів. Однак у загальному випадку термін дії ключів у значній мірі залежать від якості їхнього захисту й довжини. У загальному випадку, терміни дії більше довгих ключів більше. Аналогічно більше захищені ключі служать довше. Наприклад, ключі, що зберігаються в стійкі до злому апаратних криптоустройствах надійніше, ніж ключі, розміщені на жорсткому диску локального комп'ютера. Тому для двох ключів однієї довжини термін дії ключа, збереженого на апаратному криптографічному пристрої, звичайно більше, ніж ключа, розміщеного в програмному постачальнику CSP на жорсткому диску.

Розробка планів відновлення

Розробка детального плану відновлення дозволяє швидко повернути ЦС у робочий стан у випадку збоїв служб сертифікації або компрометації. Рекомендується протестувати складений план, щоб переконатися в його коректності. Крім того варто провести навчання співробітників, щоб бути впевненим у тім, що вони знають, як діяти відповідно до цього плану.

План відновлення повинен передбачати:

– процедури відновлення й контрольні списки для адміністраторів;

– набори засобів і службових програм відновлення;

– план дій у непередбачених обставинах.

Існує безліч причин, по яких ЦС може потерпіти збій, у тому числі поломка жорсткого диска , відмова мережної карти або вихід з ладу системної (материнської) плати . Деякі збої усуваються швидко – шляхом локалізації й виправлення джерела неполадки ЦС. Наприклад, після заміни несправної

мережної карти або системної плати для відновлення служби сертифікації досить запустити знову комп'ютер.

Жорсткий диск, що відмовив, заміняється, і ЦС відновлюється з останнього архіву. При ушкодженні або руйнуванні ЦС його відновлюють також з останнього архіву. Після цього ЦС установлюють у вихідній конфігурації й з вихідним особистим ключем і сертифікатом ЦС.

При виявленні факту компрометації ЦС треба негайно:

– відкликати сертифікат скомпрометованого ЦС;

– опублікувати новий список CRL з відкликаним сертифікатом ЦС;

– видалити скомпрометовані сертифікати ЦС зі сховища TRCA (Довірені кореневі центри сертифікації) і із всіх списків довіри (CTL);

– сповістити всіх зацікавлених користувачів і адміністраторів про компрометацію й відізвати сертифікати, які випущені скомпрометованим ЦС;

– усунути всі «лазівки», що стали причиною компрометації.

Для відновлення ієрархії центрів сертифікації варто повторно виконати розгортання нового ЦС, далі – повторно випустити всі сертифікати для користувачів, комп'ютерів і служб.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що

суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа i-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Висновок

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує систему проектування сервісів автентифікації.

Програмно-апаратні вимоги:

– Загальний обсяг ОЗП: 512 Мбайт.

– Вільний простір на жорсткому диску: 35 Мбайт.

– Операційна система Microsoft Windows 10/11.

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1

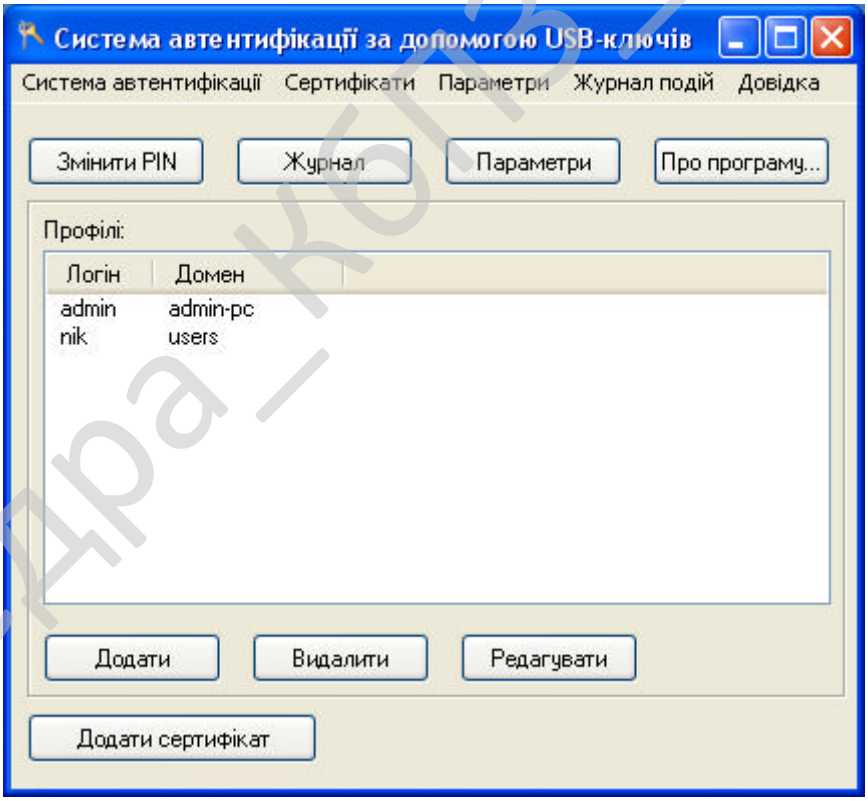

Рисунок 5.1 – Основне вікно програми

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку «Про програму…», після чого на екрані з’явиться вікно показане на рисунку 5.2.

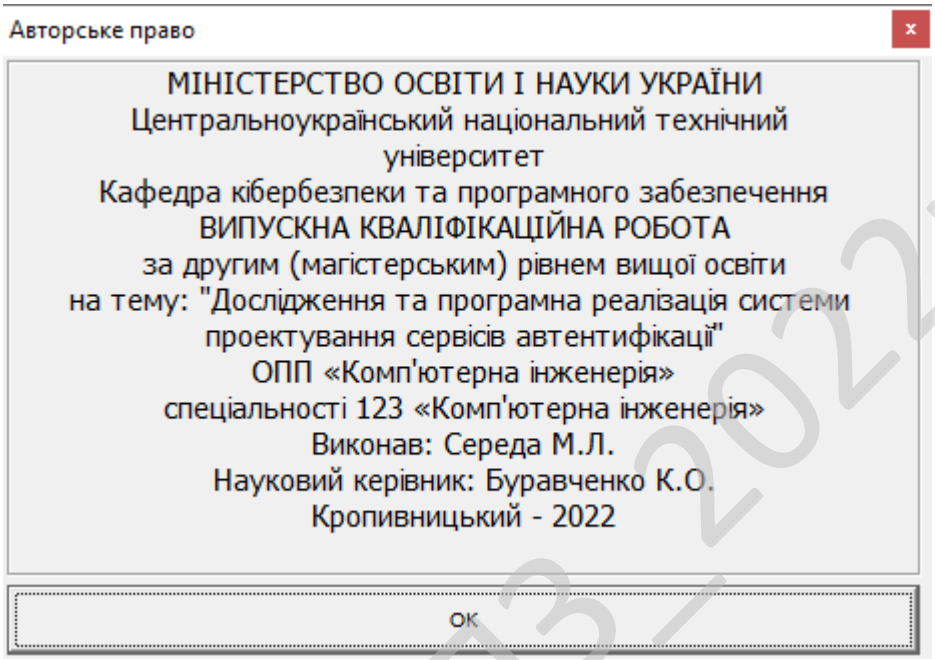

Рисунок 5.2 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи проектування сервісів автентифікації.

Метою розробки є дослідження та програмна реалізація системи проектування сервісів автентифікації.

Об'єктом дослідження є процес проектування сервісів автентифікації.

Предметом дослідження є методи проектування сервісів автентифікації.

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

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

– Удосконалено метод проектування сервісів автентифікації.

– Розроблено вітчизняний продукт проектування сервісів автентифікації, 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 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи проектування сервісів автентифікації.

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	40
3. Запланований термін розробки, днів	Fpq	60 (3 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
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
. Кількість форм вихідної інформації.	–	4
. Мова програмування (1-6)	–	4
0. Попередній досвід (1-6)	–	3
1. Гнучкість проекту ПП (1-6)	–	3
2. Детальність проекту ПП (1-6)	_	2
3. Рівень спрацьованості колективу (1-6)	_	2
4. Ступінь вимірності процесів (1-6)	_	3
5. Необхідна надійність програмного забезпечення (1-6)	_	2
6. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
7. Складність кінцевого програмного продукту (1-6)	_	2
8. Необхідний рівень забезпечення повторного використання (1-6)	_	2
9. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
0. Вимоги до швидкодії ПП (1-6)	_	2
1. Обмеження на розміри основного сховища даних (1-6)	_	2
2. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
3. Професійний рівень аналітиків (1-6)	_	2
4. Професійний рівень програмістів (1-6)	_	2
5. Постійність складу команди розробників (1-6)	_	2
6. Досвід розробки додатків (1-6)	_	2
7. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
8. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
9. Досвід роботи з програмними інструментами розробки (1-6)	–	3
0. Розробка ПО для декількох серверів одночасно (1-6)	–	2
1. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
2. Вартість ПЗ у розробника (НМА), грн	–	40000
3. Норматив додаткової зарплати, % :	$Н_д$	10
4. Норматив відрахувань у соціальні фонди, %	Нс	22
5. Норматив загальногосподарських витрат, %	Нг	15
6. Норматив витрат на освоєння нових мов програмування, %	Нп	15
7. Рівень рентабельності програмної продукції, %	Ре	55
8. 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 Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

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

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 $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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:

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

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

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot45=76 \text{ люд/день}$$

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	76	Ф 7.1-7.4
Впровадження	13	Д13
Всього	117	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{117 \cdot 1}{60\text{-}5} = 2{,}1 \text{ ставки}$$

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	5	450	7,5
Монітор	60	5	300	5
Клавіатура	30	5	150	2,5
Маніпулятор «мишка»	30	5	150	2,5
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	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	31,16

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{31,16 \cdot 3}{1,2} = 78 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=78/(60·8)=0,15 ставки

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	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	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 д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість 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	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	0,5	18200	27300
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,1	16000	100800
Інженер-електронщик	0,15	12000	5400
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	12000	18000
Всього за період розробки	$R_{сп}$=5	-	$Ф_{роб}$=214500

Розрахуємо середньоденну зарплату одного виконавця:

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

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{214500}{5 \cdot 60} = 715 \; грн.$$

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

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sу – питома площа на одне робоче місце, $м^2$; Цпл – 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

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де Цм – 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 за прайсом фірми Компбест за 06.11.22 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i5-3470: 4 ядра (4 потоки) по 3.2 – 3.6 GHz, 6mb Cache.	-
Системна плата	Intel, USB, VGA, DVI, RJ-45, audio, LGA1155	-
Відеокарта	GIGABYTE GV-N710 GeForce GT710 1 GI DDR4 64-bit D-Sub+HDMI+DVI	-
Жорсткий диск	240 SSD+HDD Seagate Barracuda 750 Gb 7200 32Mb SATAII ST3750528AS (ST3750528AS)	-
Оперативна пам’ять	DIMM 4Gb DDR3 , non-Reg., no-ECC , CL 9 (2 модулі)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 2.0, Mic+Audio, silver/black	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI 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	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

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	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot T_{пз}}{N_e}, \qquad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 715 \cdot 117/40 = 2090 \text{ грн}$$

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, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=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, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

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:

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

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

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 33,6 грн./шт., DVD-R box – 49,2 грн./шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

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.

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

$$З_М = (105 + 492 + 1702)/40=57 \text{ грн.}$$

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

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

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_п = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

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

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$C_п = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

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	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

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	–	9168
Всього капітальних витрат	–	9168

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 оплату праці	$З_р$	77299	14494
2. Витрати на електроенергію	$З_{ел}$	2099	1049
3. Витрати на амортизацію	$З_{ам}$	0	2292
Всього витрат за рік	$І$	79398	17835

Витрати на оплату праці:

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

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 800 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 800 \cdot 72 \cdot 1{,}1 \cdot 1{,}22 = 77299\ грн.$$

до

$$З_{р\ нов} = 150 \cdot 72 \cdot 1{,}1 \cdot 1{,}22 = 14494\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

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

Зел баз=·0,475·2455·1,8 = 2099 грн

Зел нов=·0,475·1227·1,8 = 1049 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	–	9168	–	2292
Всього відрахувань	-	–	9168	–	2292

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.

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \; грн.$$

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 без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 0,5 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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

$$E_{сп} = (79398 - 17835) - 0,25 \cdot 9168 = 59271 \text{ грн.}$$

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{9168}{79398 - 17835} = 0{,}15 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	59271
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,15

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

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

8.2 Аналіз умов праці програміста

Умови праці в приміщенні, в якому знаходиться робоче місце програміста є сприятливими. Приміщення обладнане автономною системою газового

опалення, основною перевагою якого є програмування режиму роботи в залежності від погодних умов, оскільки клімат є нестійким. Використовується система природної та штучної вентиляції, що забезпечує ефективну циркуляцію повітря. В кабінеті знаходяться 1 кондиціонер (CARRIER 42QCR018/38QCR018).

Засоби копіювальної техніки знаходяться на достатньо далекій відстані від робочих місць, оскільки приміщення складає 20 м^2, а у відділі налічується два працівники , тобто концентрація озону та оксиду азоту в повітрі є невисокою. Таким чином на кожного програміста приходиться 10,0 м^2 що відповідає нормам Державним санітарним правилам і нормам ДСанПіН 3.3.2.007-98 [1]. Висота стелі приміщення складає 3 метри, що також не порушує нормативні вимоги.. Прибиральники підтримують порядок в службових приміщеннях, дотримуються санітарно-гігієнічних норм по прибиранню приміщень, витирають пил, підмітають підлогу наприкінці кожного робочого дня.

В цілому потрібно відмітати застаріліть офісної техніки та відсутність клавіатур з ергономічною розкладкою та рідкокристалічних моніторів, які здійснюють менш негативний вплив на стан здоров'я працівників відділу.

Оформлення інтер'єру приміщення є відповідає вимогам з ергономіки та стимулює працівників до підвищення працездатності та зниження втоми. Стеля білого кольору створює оптичний ефект збільшення висоти приміщення, підлога пофарбована коричневим кольором, а стіни – у жовтий. Перевагами даного кольору є створення відчуття теплоти, здатність привертати увагу без додаткової втоми.

Висота столу складає 72,5 см, до того ж його можна регулювати відповідно до власних потреб. Стіл має достатній внутрішній об'єм, завдяки ширині у 73 см та висоті простору під столом – у 64 см, є достатньо важким для забезпечення стійкості. Крісла забезпечують фізіологічно раціональну позу, мають підлокітники, здатні обертатися та регулятор висоти, кута нахилу спинки й відстані спинки від краю сидіння.

В кабінеті створено оптимальні умови праці відносно температури, вологості приміщення та вентиляції.

Наприкінці аналізу небезпечних факторів праці побудуємо підсумкову таблицю 8.1.

Таблиця 8.1 – Підсумкова таблиця значень параметрів небезпечних факторів праці

Найменування параметра	Значення параметра		Нормативний документ
	Фактичне	Нормоване	
1	2	3	4
Освітленість штучна, лк	300	300	ДБН.В 2.5-28:2018 [2]
Значення КПО,%	1,0	1,1	ДБН.В 2.5-28:2018 [2]
Повітрообмін,м /год			
взимку	76	80	ДСН 3.3.6.042-99[3]
влітку	36	80	ДСН 3.3.6.042-99 [3]
Температура повітря. °С			
взимку	22	21-25	ДСанПіН 3.3.2-007-98
влітку	24	27-28	ДСанПіН 3.3.2-007-98
Відносна вологість,%			
взимку	60	<75	ДСанПіН 3.3.2-007-98
влітку	55	<60	ДСанПіН 3.3.2-007-98
Швидкість переміщення повітря, м/с			
взимку	0,16	<0,2	ДСанПіН 3.3.2-007-98
влітку	0,10	<0,2	ДСанПіН 3.3.2-007-98

Щодо вимог електробезпеки, то приміщення за небезпекою ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, без пилу, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

Для запобігання поразки електричним струмом в приміщенні відділу використовується ряд організаційно-технічних заходів: розташування проводів живлення поза зоною пересування людей; допуск до роботи електроприладів тільки тих робітників, що знайомі із технікою безпеки; використання мережних продовжувачів з вбудованими запобіжниками на 0,1 А; при ремонті обладнання персонал попереджується.

Устаткування, що працює в приміщенні живиться від мережі 220В та частотою 50Гц. Споживачами цієї напруги є також джерела штучного освітлення. Вони розташовуються на висоті 3 м, що задовольняє нормі, відповідно до якого джерела освітлення повинні розташовуватися на висоті 2,5 м від підлоги.

Проводка схована. У якості розеток для підключення устаткування застосовуються розетки з заземленим кожухом, захищеного від випадкового доторку до струмоведучих частин. Електроустаткування, що знаходиться в приміщенні відділу відноситься до установок напругою до 1000В.

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп’ютера, але тут використовуються системні блоки, що відповідають стандартові фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення.

Основні причини ураження людини електричним струмом на робочому місці:

- дотик до металевих неструмоведучих частин (корпусу, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції;
- нерегламентоване використання електричних приладів;
- відсутність інструктажу співробітників з правил електробезпеки.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ м.

Отже за результатами проведеного аналізу можна зробити висновки, що всі показники знаходяться у межах запропонованих значень

8.3 Розробка заходів пожежної безпеки

Ступінь вогнестійкості будинків приймається в залежності від їхнього призначення, категорії по вибухопожежній і пожежній небезпеці, по поверховості, площі поверху в межах пожежного відсіку згідно ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків, установок за вибухопожежною та пожежною небезпекою [4]

За критеріями по пожежній безпеці будівля відноситься до категорії Д. За особливостями функціонування установи вибухонебезпечних парів та концентратів не помічено. Будівля побудована мармуроподібного вапняка межа стійкості складає 0,5-2,5 години.

Для профілактики пожежі надзвичайно важлива правильна оцінка пожежонебезпеки будинку, визначення небезпечних факторів і обґрунтування способів і засобів пожежопопередження і захисту.

Одне з умов забезпечення пожежобезпеки – ліквідація можливих джерел запалення.

У приміщенні програмістів джерелами запалення можуть бути:

- несправне електроустаткування, несправності в електропроводці, електричних розетках і вимикачах. Для виключення виникнення пожежі з цих причин необхідно вчасно виявляти й усувати несправності, проводити плановий огляд і вчасно усувати всі несправності;
- несправні електроприлади. Необхідні міри для виключення пожежі містять у собі своєчасний ремонт електроприладів, якісне виправлення поломок, не використання несправних електроприладів;
- обігрівання приміщення електронагрівальними приладами з відкритими нагрівальними елементами. Відкриті нагрівальні поверхні можуть спричинити

пожежу, тому що в приміщенні знаходяться паперові документи, а папір – легкозаймистий предмет. З метою профілактики пожежі пропоную не використовувати відкриті обігрівальні прилади в приміщенні лабораторії;

- коротке замикання в електропроводці. З метою зменшення імовірності виникнення пожежі внаслідок короткого замикання необхідно, щоб електропроводка була схованою.

- влучення в будинок блискавки. У літній період під час грози можливе влучення блискавки внаслідок чого можливий пожежа. Щоб уникнути цього я рекомендую установити на даху будинку блискавковідвід;

- недотримання мір пожежної безпеки і паління в приміщенні також може спричиниш пожежу. Для усунення загоряння в результаті паління в приміщенні лабораторії пропоную категорично заборонити паління, а дозволити тільки в строго відведеному для цього місці.

За протипожежним режимом визначені: місця куріння, правила проїзду та стоянки транспортних засобів, порядок прибирання пилу й відходів, відключення від мережі електрообладнання у разі пожежі, огляду і закриття приміщень після закінчення операційного дня, проходження посадовими особами навчання та тестування знань з питань пожежної безпеки, проведення з програмістами протипожежйих інструктажів, організації експлуатації і обслуговування технічних засобів протипожежного захисту, проведення планово-попереджувальних ремонтів і оглядів інженерного обладнання, дії працівників при виявленні пожежі.

8.4 Розробка заходів по електробезпеці для приміщень з ПЕОМ

Згідно ДБН В.2.5 -28:2018 Державних будівельних норм ДБН В.2.5 -28:2018 [5] при проектуванні систем електропостачання, при монтажі силового електроустаткування і електричного освітлення і в будівлях і приміщеннях для ЕОМ необхідно дотримуватися вимог нормативно-технічної документації.

ПЕОМ є однофазним споживачем електроенергії, що живиться змінним струмом напругою 220В і частотою 50Гц, від мережі із заземленою нейтраллю. За засобами захисту людини від поразки електричним струмом ЕОМ повинне відповідати першому класу захисту. Захист від випадкового дотику до струмоведучих частин забезпечують конструктивні, схемно-конструктивні і експлуатаційні заходи захисту. Комплекс необхідних заходів по електробезпеці визначається, виходячи з видів електроустановки, її номінальної напруги, умов середовища, типу приміщення і доступності електроустаткуванні.

В приміщенні розміщено декілька комп'ютерів, то кабель прокладають в металевих трубах і гнучких металевих рукавах з відведеннями. Якщо ЕОМ розміщені в центрі приміщення, електромережа прокладається в каналах або під знімною підлогою в металевих трубах і гнучких металевих рукавах.

8.5 Розрахунок занулення

Мета розрахунку – визначити умови, за яких швидко вимикається пошкоджена електроустановка, тобто визначити поперечний переріз нульового дроту, який забезпечить протікання струму однофазного короткого замикання, достатнього для спрацювання максимального захисту.

Розрахунок заземлення нейтралі і повторного заземлення нульового робочого

провідника, що забезпечують безпеку дотику до занулених пристроїв,

Схема електропостачання зануленої електроустановки представлена на рисунку 8.1, де

Тр U_1/U_2 – понижуючий трансформатор масляний, схема з'єднання обмоток – зірка-зірка;

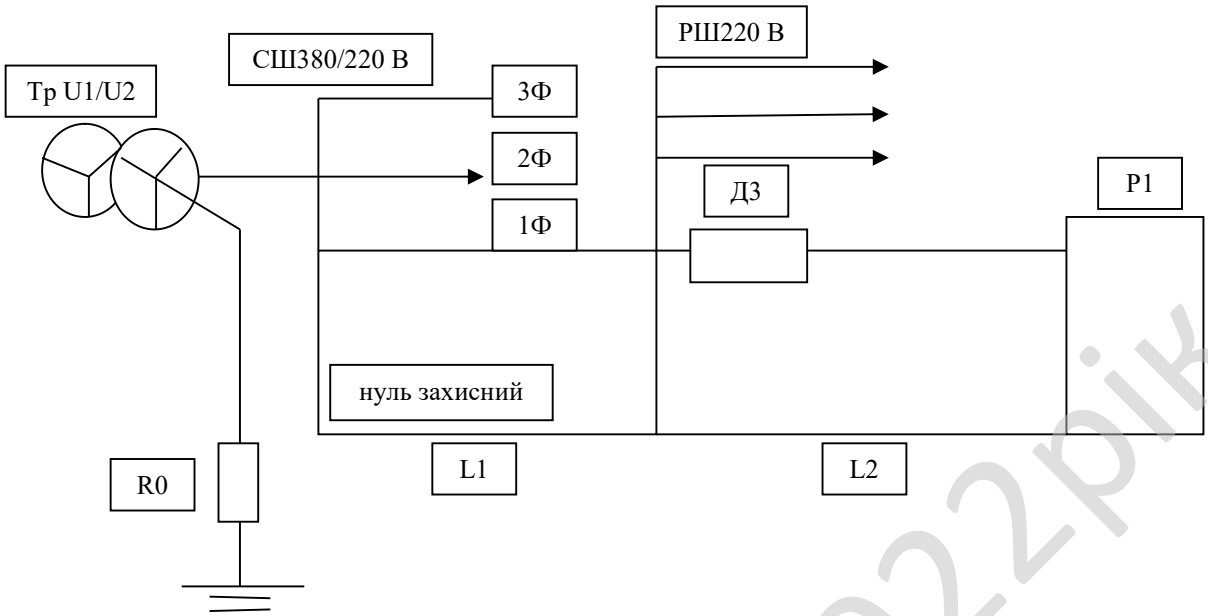

Рисунок 8.1 – Схема електропостачання зануленої електроустановки

СШ – збірна шина;

РЩ – розподільний щит;

А.З. – апарат захисту;

L_1 – довжина ділянки сіті від розподільного щита до електроустановки;

L_2 – довжина ділянки сіті від знижувального трансформатора до розподільного щита;

R_O – опір заземлення нейтральної точки;

P_1 – потужність споживача (комп'ютери, принтер. ксерокс, електроосвітлення);

Електромережа виконана як двох провідна мережа, що складається з фазного дроту і нульового захисного провідників.

$$L_1=45 \text{ м}; L_2=450 \text{ м}; P_1=2550 \text{ Вт}.$$

Матеріал жили – мідь.

Постановка задачі занулення електроустановки: визначення такого перетину нульового захисного провідника при якому струм короткого замикання

$I_{кз}$ в задане число раз До перевищить номінальний струм спрацьовування апарату захисту $I_{ном}$, Що забезпечить відключення пошкодженого споживача.

1) Вибір типу автоматичного вимикача.

1а) Визначення струму, що споживає електроустановка потужністю P_1 = 2550 Вт:

$$I_{1н} = P_1/U_ф = 2550/220 = 11{,}59 \text{ А} \quad (8.1)$$

де

$U_ф$ – фазна напруга (220 В);

1б) Визначення розрахункової величини струму спрацьовування захисного апарату:

$$I_{расч} = (K_п/K_т)I_{1н} = (3/2.5)11{,}59 = 1{,}909 \text{ А}. \quad (8.2)$$

де

$K_п$ = 3 – коефіцієнт кратності пускового струму;

$K_т$ = 2.5 – коефіцієнт тяжкості пуску електроустановки (залежить від часу пуску: t = 5 сек., пуск легкий).

1в) Вибір типу автоматичного вимикача і визначення величини струму спрацьовування апарату захисту:

$I_{ном}$ = 16 А; тип автоматичного вимикача АЕ2026.

2) Визначення струму короткого замикання фази на корпус електроустановки:

$$I_{кз} = U_ф/ Z_{пфн} \quad (8.3)$$

$Z_{пфн}$ – опір петлі фаза-нуль.

2а) Перетин фазного дроту визначається залежно від допустимого тривалого струму, способу прокладки дротів і матеріалу дротів:

Для знаходження перетину дроту, визначимо діаметр дроту d. Де d визначають по значенню струму I [А], і допустимої густини струму J [А/мм^2]. Табличне значення: J=5 [А/мм^2].

$$d=16/5=3.2 \text{ мм}^2$$

8.6 Висновки до розділу

Повсякденний контроль за дотриманням, виконанням правил пожежної безпеки по експлуатації електроприладів, кондиціонерів, вентиляційних систем, систем опалення та застосування заходів до усунення порушень, що можуть призвести до виникнення пожежі проводиться відповідальною особою. Це є основною запорукою пожежної безпеки в приміщеннях установи.

Враження електрострумом високої напруги є фактором короткочасної дії, що відрізняється не постійним погіршенням здоров'я (за винятком фізичного враження нервової системи), а разовим враженням, що приводить до тимчасового шоку чи смерті.

Для зменшення кількості випадків смертельного ураження струмом розроблено обов'язкові заходи. До цих заходів належать технічні захисні конструкції., та правила використання приладів, що є під напругою, використання заземлень корпусів обладнання та занулення струмопровідної мережі.

Отже в даному розділі магістерської роботи були визначені можливі дії установи в якій працюють програмісти, зайняті розробкою вказаного програмного забезпечення, для попередження небезпечних ситуацій відносно найбільш суттєвих факторів ризику.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи проектування сервісів автентифікації.

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 проектування сервісів автентифікації.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем проектування сервісів автентифікації.

– Досліджена система проектування сервісів автентифікації.

– На основі отриманих результатів досліджень створена програмна реалізація системи проектування сервісів автентифікації.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання проектування сервісів автентифікації.

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 Delphi 10.4.1. 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 MISTY1 .

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 59271 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,15 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Середа М.Л. Дослідження та програмна реалізація системи проектування сервісів автентифікації // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Ричард Э. Смит Аутентификация: от паролей до открытых ключей = Authentication: From Passwords to Public Keys First Edition. — М.: «Вильямс», 2002. — С. 432. — ISBN 0-201-61599-1

3. М.Э.Смид, Д.К.Бранстед. Стандарт шифрования данных: прошлое и будущее. /пер. с англ./ М., Мир, ТИИЭР.–1988.–т.76.–N5.

4. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования ГОСТ 28147–89, М., Госстандарт, 1989.

5. Б.В.Березин, П.В.Дорошкевич. Цифровая подпись на основе традиционной криптографии//Защита информации, вып.2.,М.: МП "Ирбис-II",1992.

6. W.Diffie,M.E.Hellman. New Directions in cryptography// IEEE Trans. Inform. Theory, IT-22, vol 6 (Nov. 1976), pp. 644-654.

7. У.Диффи. Первые десять лет криптографии с открытым ключом. /пер. с англ./ М., Мир, ТИИЭР.–1988.–т.76.–N5.

8. 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).**

9. 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**).

10. 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).**

11. 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).**

12. 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).**

13. 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).**

14. 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).**

15. 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)**.

16. 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)**.

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

18. 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)**.

19. 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)**.

20. 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)**.

21. 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)**.

22. 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. S**pringer**, Cham. 2021. pp 557-587. **(Scopus)**.

23. 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)**.

24. 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)**.

25. 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)**.

26. 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)**.

27. 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)**.

28. 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)**.

29. 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)**.

30. 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)**.

31. 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)**.

32. 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)**.

33. 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).

34. 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).

35. 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).

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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

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

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

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

52. 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

53. 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

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

55. 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

56. 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

57. 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

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

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

60. 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

					ВКРМ-123.22.0019.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Середа М.Л.			Дослідження та програмна реалізація системи проектування сервісів автентифікації	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Буравченко К.О.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи проектування сервісів автентифікації.

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. № 19-13 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 проектування сервісів автентифікації.

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 проектування сервісів автентифікації;

– 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 Delphi 10.4.1.

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 повинен бути розглянутий аналіз умов праці програміста.

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 – 129 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 21.12.2022 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

_____ Буравченко К.О.

Дослідження та програмна реалізація

системи проектування сервісів автентифікації

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 48

Літера: РП

Кропивницький – 2022 року

Файл FleshKey.pas – робота з флеш-ключами

```
unit FleshKey;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
FGIntRSA, FGIntRSA, FGInt, FGIntPrimeGeneration
Dialogs, StdCtrls,tlhelp32,buttons,registry, CrKey, about;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormPaint2(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  stop:boolean;
  pass:string;

procedure BlockInput; external user32;

implementation

uses Unit2;

{$R *.dfm}

function GetpidByname(sl:string):cardinal;
var
  h: HWND;
  snap: tprocessentry32;
begin
result:=0;
  h := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, cardinal(-1));
  snap.dwSize := SizeOf(ProcessEntry32);
  if process32first(h, snap) = true then
  repeat
             if lowercase(sl)=lowercase(snap.szExeFile)then
             begin
                  result:=snap.th32ProcessID;
                  break;
             end;
  until process32next(h,snap)<>true;
  closehandle(h);
end;
// блокування комп'ютера
procedure Block(s:boolean);
var
p:cardinal;
begin
```

```
if s=true then
begin
asm
   push 1
   call blockinput;
end;
p:=getpidbyname('taskmgr.exe');
if p<>0 then
   begin
        p:=Openprocess(PROCESS_TERMINATE,true,p);
        terminateprocess(p,1);
        closehandle(p);
   end;
setwindowpos(form1.handle,HWND_TOPMOST,0,0,0,0,swp_nomove or swp_nosize);
end else
begin
asm
   push 0
   call blockinput;
end;
end;
end;
//Визначення кількості дисків
procedure Scan;
var
j:word;
a:array[1..512]of byte;
s:string;
readed:cardinal;
f:cardinal;
begin
     for j:=ord('A')  to ord('Z') do
     if getdrivetype(pchar(chr(j)+':\'))=DRIVE_REMOVABLE then
     begin

f:=createfile(pchar('\\.\'+chr(j)+':'),GENERIC_READ,FILE_SHARE_READ,nil,OPEN_EXI
STING,FILE_FLAG_RANDOM_ACCESS,0);
          if f<>INVALID_HANDLE_VALUE then
          begin
          setfilepointer(f,512*4,nil,0);
          readfile(f,a,sizeof(a),readed,nil);
          readed:=1;
          while (a[readed]<>0) do
          begin
               s:=s+chr(a[readed]);
               inc(readed);
          end;
          if s=pass then
            begin
                block(false);
                winexec(pchar('"'+paramstr(0)+'" -d '+chr(j)),sw_show);
                exitprocess(1);
            end;
         closehandle(f);
         end;
     end;
end;
// Розблокування комп’ютера
procedure DoBlock;
begin
stop:=false;
while stop<>true do
      begin
           block(true);
           scan;
           application.ProcessMessages;
      end;
end;
// Обробка натискання клавіш
```

```
procedure TForm1.Button1Click(Sender: TObject);
begin
showcursor(false);
onpaint:=formpaint2;
left:=0;
top:=0;
button1.Hide;
button2.Hide;
button3.Hide;
showcursor(false);
color:=clBlack;
width:=screen.Width;
height:=screen.Height;
doblock;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
exitprocess(1);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
with TRegistry.Create(KEY_READ)do
begin
     rootkey:=HKEY_CURRENT_USER;
     if openkey('software\microsoft\windows\currentversion\Run',false)=true then
        form2.CheckBox1.Checked:=valueexists('fdcl');
     closekey;
     free;
end;
form2.Show;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
buttons.DrawButtonFace(canvas,clientrect,3,bsNew,true,false,false);
end;
// Створення головного вікна
procedure TForm1.FormCreate(Sender: TObject);
begin
left:=screen.Width-width;
top:=screen.WorkAreaHeight-height;
with TRegistry.Create(KEY_READ)do
begin
     rootkey:=HKEY_CURRENT_USER;
     if openkey('SOFTWARE',false)=true then
     if valueexists('fdcl')then
        pass:=readstring('fdcl');
     closekey;
     free;
end;
end;
// активізація головного вікна
procedure TForm1.FormActivate(Sender: TObject);
begin
showwindow(application.Handle,sw_hide);
if paramstr(1)='-lock'then button1.Click;
if paramstr(1)='-d'then
begin
   while windows.GetDriveType(pchar(paramstr(2)+':\'))=DRIVE_REMOVABLE do
         application.ProcessMessages;
   button1.Click;
end;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
frm_about.Show;
```

```
end;

end.
```

Файл CrKey.pas – створення ключа

```
unit CrKey;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls,registry;

type
  TForm2 = class(TForm)
    drives: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    CheckBox1: TCheckBox;
    Button3: TButton;
    procedure drivesDropDown(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}
// Визначення дисків
procedure TForm2.drivesDropDown(Sender: TObject);
var
j:cardinal;
begin
drives.Clear;
for j:=ord('A')  to ord('Z') do
    if Getdrivetype(pchar(chr(j)+':\'))=DRIVE_REMOVABLE then
       drives.Items.Add(chr(j));
end;

procedure TForm2.Button1Click(Sender: TObject);
var
f:cardinal;
s:string;
w:cardinal;
a:array[1..512]of byte;
begin
with TREgistry.Create(KEY_WRITE)do
begin
     rootkey:=HKEY_CURRENT_USER;
     if openkey('software\microsoft\windows\currentversion\Run',false)=true then
     begin
          if checkbox1.Checked=true then
             writestring('fdcl','"'+paramstr(0)+'" -lock')else
             deletevalue('fdcl');
     end;
     closekey;
     free;
end;
if edit1.Text=''then
```

```
   begin
        showmessage('Введіть PIN');
        exit;
   end;
if drives.Text<>''then
   begin
   RSAEncrypt(drives.Text, e, n, st);

f:=createfile(pchar('\\.\'+drives.Text+':'),GENERIC_WRITE,FILE_SHARE_WRITE,nil,O
PEN_EXISTING,FILE_FLAG_RANDOM_ACCESS,0);
        if f<>INVALID_HANDLE_VALUE then
           begin
                setfilepointer(f,512*4,nil,0);
                for w:=1  to 512 do
                    a[w]:=0;
                s:=edit1.Text;
                for w:=1 to length(s)do
                    a[w]:=ord(s[w]);
                writefile(f,a,sizeof(a),w,nil);
                with TRegistry.Create(KEY_WRITE)do
                begin
                     rootkey:=HKEY_CURRENT_USER;
                     if openkey('SOFTWARE',false)=true then
                        writestring('fdcl',s);
                     closekey;
                     free;
                end;
           closehandle(f);
           end;
   end;
close;
winexec(pchar(paramstr(0)),sw_show);
exitprocess(1);
end;

procedure TForm2.Button2Click(Sender: TObject);
begin
close;
end;

procedure TForm2.Button3Click(Sender: TObject);
var
j:cardinal;
s:string;
begin
randomize;
for j:=1  to edit1.MaxLength do
    s:=s+chr(random(126-32)+32);
edit1.Text:=s;
end;

end.
```

Файл FGIntRSA.pas – алгоритм шифрування RSA

```
Unit FGIntRSA;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);


Implementation


{$H+}

// Шифруємо рядок алгоритмом RSA, P^exp mod modb = E

Procedure RSAEncrypt(P : String; Var exp, modb : TFGInt; Var E : String);
Var
   i, j, modbits : longint;
   PGInt, temp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(P, tempstr1);
   tempstr1 := '111' + tempstr1;
   j := modbits - 1;
   While (length(tempstr1) Mod j) <> 0 Do tempstr1 := '0' + tempstr1;

   j := length(tempstr1) Div (modbits - 1);
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits - 1);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, PGInt);
      delete(tempstr1, 1, modbits - 1);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
FGIntMontgomeryModExp(PGInt, exp, modb, temp);
      FGIntDestroy(PGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;

   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1,
1);
   ConvertBase2To256(tempstr2, E);
   FGIntDestroy(zero);
End;


Дешифруємо рядок алгоритмом RSA, E^exp mod modb = D
// modb = p*q, d_p*e mod (p-1) = 1
// d_q*e mod (q-1)

Procedure RSADecrypt(E : String; Var exp, modb, d_p, d_q, p, q : TFGInt; Var D :
String);
```

```
Var
   i, j, modbits : longint;
   EGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp, zero : TFGInt;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToFGInt('0', zero);
   FGIntToBase2String(modb, tempstr1);
   modbits := length(tempstr1);
   convertBase256to2(E, tempstr1);
   While copy(tempstr1, 1, 1) = '0' Do delete(tempstr1, 1, 1);
   While (length(tempstr1) Mod modbits) <> 0 Do tempstr1 := '0' + tempstr1;
   If exp.Number = Nil Then
   Begin
      FGIntModInv(q, p, temp1);
      FGIntMul(q, temp1, qqinvp);
      FGIntDestroy(temp1);
      FGIntModInv(p, q, temp1);
      FGIntMul(p, temp1, ppinvq);
      FGIntDestroy(temp1);
   End;

   j := length(tempstr1) Div modbits;
   tempstr2 := '';
   For i := 1 To j Do
   Begin
      tempstr3 := copy(tempstr1, 1, modbits);
      While (copy(tempstr3, 1, 1) = '0') And (length(tempstr3) > 1) Do
delete(tempstr3, 1, 1);
      Base2StringToFGInt(tempstr3, EGInt);
      delete(tempstr1, 1, modbits);
      If tempstr3 = '0' Then FGIntCopy(zero, temp) Else
      Begin
         If exp.Number <> Nil Then FGIntMontgomeryModExp(EGInt, exp, modb, temp)
Else
         Begin
            FGIntMontgomeryModExp(EGInt, d_p, p, temp1);
            FGIntMul(temp1, qqinvp, temp3);
            FGIntCopy(temp3, temp1);
            FGIntMontgomeryModExp(EGInt, d_q, q, temp2);
            FGIntMul(temp2, ppinvq, temp3);
            FGIntCopy(temp3, temp2);
            FGIntAddMod(temp1, temp2, modb, temp);
            FGIntDestroy(temp1);
            FGIntDestroy(temp2);
         End;
      End;
      FGIntDestroy(EGInt);
      tempstr3 := '';
      FGIntToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      FGIntdestroy(temp);
   End;

   If exp.Number = Nil Then
   Begin
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   While (Not (copy(tempstr2, 1, 3) = '111')) And (length(tempstr2) > 3) Do
delete(tempstr2, 1, 1);
   delete(tempstr2, 1, 3);
   ConvertBase2To256(tempstr2, D);
   FGIntDestroy(zero);
End;


// підписуємо рядок RSA, M^d mod n = S
```

```
// n = p*q, dp*e mod (p-1) = 1
// dq*e mod (q-1)


Procedure RSASign(M : String; Var d, n, dp, dq, p, q : TFGInt; Var S : String);
Var
   MGInt, SGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp : TFGInt;
Begin
   Base256StringToFGInt(M, MGInt);
   If d.Number <> Nil Then FGIntMontgomeryModExp(MGInt, d, n, SGInt) Else
   Begin
      FGIntModInv(p, q, temp);
      FGIntMul(p, temp, ppinvq);
      FGIntDestroy(temp);
      FGIntModInv(q, p, temp);
      FGIntMul(q, temp, qqinvp);
      FGIntDestroy(temp);
      FGIntMontgomeryModExp(MGInt, dp, p, temp1);
      FGIntMul(temp1, qqinvp, temp2);
      FGIntCopy(temp2, temp1);
      FGIntMontgomeryModExp(MGInt, dq, q, temp2);
      FGIntMul(temp2, ppinvq, temp3);
      FGIntCopy(temp3, temp2);
      FGIntAddMod(temp1, temp2, n, SGInt);
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
      FGIntDestroy(ppinvq);
      FGIntDestroy(qqinvp);
   End;
   FGIntToBase256String(SGInt, S);
   FGIntDestroy(MGInt);
   FGIntDestroy(SGInt);
End;


// Перевіряємо правильність алгоритму RSA,
// Якщо M = S^e mod n тоді ok:=true інакше ok:=false

Procedure RSAVerify(M, S : String; Var e, n : TFGInt; Var valid : boolean);
Var
   MGInt, SGInt, temp : TFGInt;
Begin
   Base256StringToFGInt(S, SGInt);
   Base256StringToFGInt(M, MGInt);
   FGIntMod(MGInt, n, temp);
   FGIntCopy(temp, MGInt);
   FGIntMontgomeryModExp(SGInt, e, n, temp);
   FGIntCopy(temp, SGInt);
   valid := (FGIntCompareAbs(SGInt, MGInt) = Eq);
   FGIntDestroy(SGInt);
   FGIntDestroy(MGInt);
End;

End.
```

Файл FGIntRSA.PAS - генерація сертифікату та ключа для алгоритму RSA

```
Unit FGIntPrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Procedure FGIntDestroy(Var FGInt : TFGInt);
Procedure PrimeSearch(Var GInt : TFGInt);

Implementation
// Масив простих чисел
Var   primes : Array[1..1228] Of integer =
      (3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
      131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223, 227, 229, 233, 239, 241, 251,
      257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347,
349, 353, 359, 367, 373, 379, 383, 389,
      397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479,
487, 491, 499, 503, 509, 521, 523, 541,
      547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653, 659, 661, 673, 677,
      683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839,
      853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997, 1009,
      1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,
1091, 1093, 1097, 1103, 1109, 1117, 1123,
      1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223,
1229, 1231, 1237, 1249, 1259, 1277, 1279,
      1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367,
1373, 1381, 1399, 1409, 1423, 1427, 1429,
      1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493,
1499, 1511, 1523, 1531, 1543, 1549, 1553,
      1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621,
1627, 1637, 1657, 1663, 1667, 1669, 1693,
      1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783,
1787, 1789, 1801, 1811, 1823, 1831, 1847,
      1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933,
1949, 1951, 1973, 1979, 1987, 1993, 1997,
      1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083,
2087, 2089, 2099, 2111, 2113, 2129, 2131,
      2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239,
2243, 2251, 2267, 2269, 2273, 2281, 2287,
      2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377,
2381, 2383, 2389, 2393, 2399, 2411, 2417,
      2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
2543, 2549, 2551, 2557, 2579, 2591, 2593,
      2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687,
2689, 2693, 2699, 2707, 2711, 2713, 2719,
      2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803,
2819, 2833, 2837, 2843, 2851, 2857, 2861,
      2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969,
2971, 2999, 3001, 3011, 3019, 3023, 3037,
      3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163,
3167, 3169, 3181, 3187, 3191, 3203, 3209,
      3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313,
3319, 3323, 3329, 3331, 3343, 3347, 3359,
      3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463,
3467, 3469, 3491, 3499, 3511, 3517, 3527,
```

3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607,
3613, 3617, 3623, 3631, 3637, 3643, 3659,
3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761,
3767, 3769, 3779, 3793, 3797, 3803, 3821,
3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917,
3919, 3923, 3929, 3931, 3943, 3947, 3967,
3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073,
4079, 4091, 4093, 4099, 4111, 4127, 4129,
4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231,
4241, 4243, 4253, 4259, 4261, 4271, 4273,
4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397,
4409, 4421, 4423, 4441, 4447, 4451, 4457,
4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561,
4567, 4583, 4591, 4597, 4603, 4621, 4637,
4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723,
4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909,
4919, 4931, 4933, 4937, 4943, 4951, 4957,
4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039,
5051, 5059, 5077, 5081, 5087, 5099, 5101,
5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227,
5231, 5233, 5237, 5261, 5273, 5279, 5281,
5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407,
5413, 5417, 5419, 5431, 5437, 5441, 5443,
5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531,
5557, 5563, 5569, 5573, 5581, 5591, 5623,
5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701,
5711, 5717, 5737, 5741, 5743, 5749, 5779,
5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857,
5861, 5867, 5869, 5879, 5881, 5897, 5903,
5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047,
6053, 6067, 6073, 6079, 6089, 6091, 6101,
6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211,
6217, 6221, 6229, 6247, 6257, 6263, 6269,
6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353,
6359, 6361, 6367, 6373, 6379, 6389, 6397,
6421, 6427, 6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551,
6553, 6563, 6569, 6571, 6577, 6581, 6599,
6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703,
6709, 6719, 6733, 6737, 6761, 6763, 6779,
6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869,
6871, 6883, 6899, 6907, 6911, 6917, 6947,
6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019,
7027, 7039, 7043, 7057, 7069, 7079, 7103,
7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213,
7219, 7229, 7237, 7243, 7247, 7253, 7283,
7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417,
7433, 7451, 7457, 7459, 7477, 7481, 7487,
7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561,
7573, 7577, 7583, 7589, 7591, 7603, 7607,
7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717,
7723, 7727, 7741, 7753, 7757, 7759, 7789,
7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901,
7907, 7919, 7927, 7933, 7937, 7949, 7951,
7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089,
8093, 8101, 8111, 8117, 8123, 8147, 8161,
8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311,
8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431,
8443, 8447, 8461, 8467, 8501, 8513, 8521,
8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627,
8629, 8641, 8647, 8663, 8669, 8677, 8681,
8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761,
8779, 8783, 8803, 8807, 8819, 8821, 8831,
8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941,
8951, 8963, 8969, 8971, 8999, 9001, 9007,
9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127,
9133, 9137, 9151, 9157, 9161, 9173, 9181,

```
      9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283,
9293, 9311, 9319, 9323, 9337, 9341, 9343,
      9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437,
9439, 9461, 9463, 9467, 9473, 9479, 9491,
      9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623,
9629, 9631, 9643, 9649, 9661, 9677, 9679,
      9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787,
9791, 9803, 9811, 9817, 9829, 9833, 9839,
      9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941,
9949, 9967, 9973);
//Додавання великих чисел
{$H+}
Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
Var
   i, size1, size2, size : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   If size1 < size2 Then FGIntAdd(FGInt2, FGInt1, Sum) Else
   Begin
      If FGInt1.Sign = FGInt2.Sign Then
      Begin
         Sum.Sign := FGInt1.Sign;
         SetLength(Sum.Number, size1 + 2);
         rest := 0;
         For i := 1 To size2 Do
         Begin
            Trest := FGInt1.Number[i] + FGInt2.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         For i := (size2 + 1) To size1 Do
         Begin
            Trest := FGInt1.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         size := size1 + 1;
         Sum.Number[0] := size;
         Sum.Number[size] := rest;
         While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
         If Sum.Number[0] > size Then SetLength(Sum.Number, size + 1);
         Sum.Number[0] := size;
      End
      Else
      Begin
         If FGIntCompareAbs(FGInt2, FGInt1) = Lt Then FGIntAdd(FGInt2, FGInt1,
Sum)
         Else
         Begin
            SetLength(Sum.Number, size1 + 1);
            rest := 0;
            For i := 1 To size2 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] - FGInt2.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            For i := (size2 + 1) To size1 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            size := size1;
            While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
            If size < size1 Then
```

```
            Begin
               SetLength(Sum.Number, size + 1);
            End;
            Sum.Number[0] := size;
            Sum.Sign := FGInt1.Sign;
         End;
      End;
   End;
End;

Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Begin
   FGInt2.Sign := FGInt1.Sign;
   FGInt2.Number := Nil;
   FGInt2.Number := Copy(FGInt1.Number, 0, FGInt1.Number[0] + 1);
End;

Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Var
   j : int64;
   i : integer;
Begin
   If ((FGInt.Number[1] Mod 2) = 0) Then ok := false
   Else
   Begin
      i := 0;
      ok := true;
      While ok And (i < 1228) Do
      Begin
         i := i + 1;
         FGIntmodbyint(FGInt, primes[i], j);
         If j = 0 Then ok := false;
      End;
   End;
End;
// Перевірка на простоту числа
Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Begin
   FGIntTrialdiv9999(FGIntp, ok);
   If ok Then FGIntRabinMiller(FGIntp, nrRMtests, ok);
End;

Procedure FGIntDestroy(Var FGInt : TFGInt);
Begin
   FGInt.Number := Nil;
End;
//Пошук простих чисел
Procedure PrimeSearch(Var GInt : TFGInt);
Var
   temp, two : TFGInt;
   ok : Boolean;
Begin
   If (GInt.Number[1] Mod 2) = 0 Then GInt.Number[1] := GInt.Number[1] + 1;
   Base10StringToFGInt('2', two);
   ok := false;
   While Not ok Do
   Begin
      FGIntAdd(GInt, two, temp);
      FGIntCopy(temp, GInt);
      FGIntPrimeTest(GInt, 4, ok);
   End;
   FGIntDestroy(two);
End;
End.
```

Файл FGInt.pas – робота з великими числами для алгоритму RSA

```
Unit FGInt;

Interface

Uses Windows, SysUtils, Controls, Math;

Type
   TCompare = (Lt, St, Eq, Er);
   TSign = (negative, positive);
   TFGInt = Record
      Sign : TSign;
      Number : Array Of int64;
   End;
// Визначення основних функцій для роботи з великими числами
Procedure zeronetochar8(Var g : char; Const x : String);
Procedure zeronetochar6(Var g : integer; Const x : String);
Procedure initialize8(Var trans : Array Of String);
Procedure initialize6(Var trans : Array Of String);
Procedure initialize6PGP(Var trans : Array Of String);
Procedure ConvertBase256to64(Const str256 : String; Var str64 : String);
Procedure ConvertBase64to256(Const str64 : String; Var str256 : String);
Procedure ConvertBase256to2(Const str256 : String; Var str2 : String);
Procedure ConvertBase64to2(Const str64 : String; Var str2 : String);
Procedure ConvertBase2to256(str2 : String; Var str256 : String);
Procedure ConvertBase2to64(str2 : String; Var str64 : String);
Procedure ConvertBase256StringToHexString(Str256 : String; Var HexStr : String);
Procedure ConvertHexStringToBase256String(HexStr : String; Var Str256 : String);
Procedure PGPConvertBase256to64(Var str256, str64 : String);
Procedure PGPConvertBase64to256(str64 : String; Var str256 : String);
Procedure PGPConvertBase64to2(str64 : String; Var str2 : String);
Procedure Base10StringToFGInt(Base10 : String; Var FGInt : TFGInt);
Procedure FGIntToBase10String(Const FGInt : TFGInt; Var Base10 : String);
Procedure FGIntDestroy(Var FGInt : TFGInt);
Function FGIntCompareAbs(Const FGInt1, FGInt2 : TFGInt) : TCompare;
Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
Procedure FGIntChangeSign(Var FGInt : TFGInt);
Procedure FGIntSub(Var FGInt1, FGInt2, dif : TFGInt);
Procedure FGIntMulByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64);
Procedure FGIntMulByIntbis(Var FGInt : TFGInt; by : int64);
Procedure FGIntDivByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64; Var
modres : int64);
Procedure FGIntDivByIntBis(Var FGInt : TFGInt; by : int64; Var modres : int64);
Procedure FGIntModByInt(Const FGInt : TFGInt; by : int64; Var modres : int64);
Procedure FGIntAbs(Var FGInt : TFGInt);
Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Procedure FGIntShiftLeft(Var FGInt : TFGInt);
Procedure FGIntShiftRight(Var FGInt : TFGInt);
Procedure FGIntShiftRightBy31(Var FGInt : TFGInt);
Procedure FGIntAddBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Procedure FGIntSubBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Procedure FGIntMul(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt);
Procedure FGIntSquare(Const FGInt : TFGInt; Var Square : TFGInt);
Procedure FGIntToBase2String(Const FGInt : TFGInt; Var S : String);
Procedure Base2StringToFGInt(S : String; Var FGInt : TFGInt);
Procedure FGIntToBase256String(Const FGInt : TFGInt; Var str256 : String);
Procedure Base256StringToFGInt(str256 : String; Var FGInt : TFGInt);
Procedure PGPMPIToFGInt(PGPMPI : String; Var FGInt : TFGInt);
Procedure FGIntToPGPMPI(FGInt : TFGInt; Var PGPMPI : String);
Procedure FGIntExp(Const FGInt, exp : TFGInt; Var res : TFGInt);
Procedure FGIntFac(Const FGInt : TFGInt; Var res : TFGInt);
Procedure FGIntShiftLeftBy31(Var FGInt : TFGInt);
Procedure FGIntDivMod(Var FGInt1, FGInt2, QFGInt, MFGInt : TFGInt);
Procedure FGIntDiv(Var FGInt1, FGInt2, QFGInt : TFGInt);
Procedure FGIntMod(Var FGInt1, FGInt2, MFGInt : TFGInt);
Procedure FGIntSquareMod(Var FGInt, Modb, FGIntSM : TFGInt);
```

```
Procedure FGIntAddMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Procedure FGIntMulMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Procedure FGIntModExp(Var FGInt, exp, modb, res : TFGInt);
Procedure FGIntModBis(Const FGInt : TFGInt; Var FGIntOut : TFGInt; b : longint;
head : int64);
Procedure FGIntMulModBis(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt; b :
longint; head : int64);
Procedure FGIntMontgomeryMod(Const GInt, base, baseInv : TFGInt; Var MGInt :
TFGInt; b : longint; head : int64);
Procedure FGIntMontgomeryModExp(Var FGInt, exp, modb, res : TFGInt);
Procedure FGIntGCD(Const FGInt1, FGInt2 : TFGInt; Var GCD : TFGInt);
Procedure FGIntLCM(Const FGInt1, FGInt2 : TFGInt; Var LCM : TFGInt);
Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Procedure FGIntRandom1(Var Seed, RandomFGInt : TFGInt);
Procedure FGIntRabinMiller(Var FGIntp : TFGInt; nrtest : integer; Var ok :
boolean);
Procedure FGIntBezoutBachet(Var FGInt1, FGInt2, a, b : TFGInt);
Procedure FGIntModInv(Const FGInt1, base : TFGInt; Var Inverse : TFGInt);
Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Procedure FGIntLegendreSymbol(Var a, p : TFGInt; Var L : integer);
Procedure FGIntSquareRootModP(Square, Prime : TFGInt; Var SquareRoot : TFGInt);



Implementation
// Масив простих чисел
Var
   primes : Array[1..1228] Of integer =
      (3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
      131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223, 227, 229, 233, 239, 241, 251,
      257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347,
349, 353, 359, 367, 373, 379, 383, 389,
      397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479,
487, 491, 499, 503, 509, 521, 523, 541,
      547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631,
641, 643, 647, 653, 659, 661, 673, 677,
      683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
797, 809, 811, 821, 823, 827, 829, 839,
      853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997, 1009,
      1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087,
1091, 1093, 1097, 1103, 1109, 1117, 1123,
      1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223,
1229, 1231, 1237, 1249, 1259, 1277, 1279,
      1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367,
1373, 1381, 1399, 1409, 1423, 1427, 1429,
      1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493,
1499, 1511, 1523, 1531, 1543, 1549, 1553,
      1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621,
1627, 1637, 1657, 1663, 1667, 1669, 1693,
      1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783,
1787, 1789, 1801, 1811, 1823, 1831, 1847,
      1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933,
1949, 1951, 1973, 1979, 1987, 1993, 1997,
      1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083,
2087, 2089, 2099, 2111, 2113, 2129, 2131,
      2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239,
2243, 2251, 2267, 2269, 2273, 2281, 2287,
      2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377,
2381, 2383, 2389, 2393, 2399, 2411, 2417,
      2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539,
2543, 2549, 2551, 2557, 2579, 2591, 2593,
      2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687,
2689, 2693, 2699, 2707, 2711, 2713, 2719,
      2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803,
2819, 2833, 2837, 2843, 2851, 2857, 2861,
```

2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969,
2971, 2999, 3001, 3011, 3019, 3023, 3037,
3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163,
3167, 3169, 3181, 3187, 3191, 3203, 3209,
3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313,
3319, 3323, 3329, 3331, 3343, 3347, 3359,
3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463,
3467, 3469, 3491, 3499, 3511, 3517, 3527,
3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607,
3613, 3617, 3623, 3631, 3637, 3643, 3659,
3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761,
3767, 3769, 3779, 3793, 3797, 3803, 3821,
3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917,
3919, 3923, 3929, 3931, 3943, 3947, 3967,
3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073,
4079, 4091, 4093, 4099, 4111, 4127, 4129,
4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231,
4241, 4243, 4253, 4259, 4261, 4271, 4273,
4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397,
4409, 4421, 4423, 4441, 4447, 4451, 4457,
4463, 4481, 4483, 4493, 4507, 4513, 4517, 4519, 4523, 4547, 4549, 4561,
4567, 4583, 4591, 4597, 4603, 4621, 4637,
4639, 4643, 4649, 4651, 4657, 4663, 4673, 4679, 4691, 4703, 4721, 4723,
4729, 4733, 4751, 4759, 4783, 4787, 4789,
4793, 4799, 4801, 4813, 4817, 4831, 4861, 4871, 4877, 4889, 4903, 4909,
4919, 4931, 4933, 4937, 4943, 4951, 4957,
4967, 4969, 4973, 4987, 4993, 4999, 5003, 5009, 5011, 5021, 5023, 5039,
5051, 5059, 5077, 5081, 5087, 5099, 5101,
5107, 5113, 5119, 5147, 5153, 5167, 5171, 5179, 5189, 5197, 5209, 5227,
5231, 5233, 5237, 5261, 5273, 5279, 5281,
5297, 5303, 5309, 5323, 5333, 5347, 5351, 5381, 5387, 5393, 5399, 5407,
5413, 5417, 5419, 5431, 5437, 5441, 5443,
5449, 5471, 5477, 5479, 5483, 5501, 5503, 5507, 5519, 5521, 5527, 5531,
5557, 5563, 5569, 5573, 5581, 5591, 5623,
5639, 5641, 5647, 5651, 5653, 5657, 5659, 5669, 5683, 5689, 5693, 5701,
5711, 5717, 5737, 5741, 5743, 5749, 5779,
5783, 5791, 5801, 5807, 5813, 5821, 5827, 5839, 5843, 5849, 5851, 5857,
5861, 5867, 5869, 5879, 5881, 5897, 5903,
5923, 5927, 5939, 5953, 5981, 5987, 6007, 6011, 6029, 6037, 6043, 6047,
6053, 6067, 6073, 6079, 6089, 6091, 6101,
6113, 6121, 6131, 6133, 6143, 6151, 6163, 6173, 6197, 6199, 6203, 6211,
6217, 6221, 6229, 6247, 6257, 6263, 6269,
6271, 6277, 6287, 6299, 6301, 6311, 6317, 6323, 6329, 6337, 6343, 6353,
6359, 6361, 6367, 6373, 6379, 6389, 6397,
6421, 6427, 6449, 6451, 6469, 6473, 6481, 6491, 6521, 6529, 6547, 6551,
6553, 6563, 6569, 6571, 6577, 6581, 6599,
6607, 6619, 6637, 6653, 6659, 6661, 6673, 6679, 6689, 6691, 6701, 6703,
6709, 6719, 6733, 6737, 6761, 6763, 6779,
6781, 6791, 6793, 6803, 6823, 6827, 6829, 6833, 6841, 6857, 6863, 6869,
6871, 6883, 6899, 6907, 6911, 6917, 6947,
6949, 6959, 6961, 6967, 6971, 6977, 6983, 6991, 6997, 7001, 7013, 7019,
7027, 7039, 7043, 7057, 7069, 7079, 7103,
7109, 7121, 7127, 7129, 7151, 7159, 7177, 7187, 7193, 7207, 7211, 7213,
7219, 7229, 7237, 7243, 7247, 7253, 7283,
7297, 7307, 7309, 7321, 7331, 7333, 7349, 7351, 7369, 7393, 7411, 7417,
7433, 7451, 7457, 7459, 7477, 7481, 7487,
7489, 7499, 7507, 7517, 7523, 7529, 7537, 7541, 7547, 7549, 7559, 7561,
7573, 7577, 7583, 7589, 7591, 7603, 7607,
7621, 7639, 7643, 7649, 7669, 7673, 7681, 7687, 7691, 7699, 7703, 7717,
7723, 7727, 7741, 7753, 7757, 7759, 7789,
7793, 7817, 7823, 7829, 7841, 7853, 7867, 7873, 7877, 7879, 7883, 7901,
7907, 7919, 7927, 7933, 7937, 7949, 7951,
7963, 7993, 8009, 8011, 8017, 8039, 8053, 8059, 8069, 8081, 8087, 8089,
8093, 8101, 8111, 8117, 8123, 8147, 8161,
8167, 8171, 8179, 8191, 8209, 8219, 8221, 8231, 8233, 8237, 8243, 8263,
8269, 8273, 8287, 8291, 8293, 8297, 8311,
8317, 8329, 8353, 8363, 8369, 8377, 8387, 8389, 8419, 8423, 8429, 8431,
8443, 8447, 8461, 8467, 8501, 8513, 8521,

```
      8527, 8537, 8539, 8543, 8563, 8573, 8581, 8597, 8599, 8609, 8623, 8627,
8629, 8641, 8647, 8663, 8669, 8677, 8681,
      8689, 8693, 8699, 8707, 8713, 8719, 8731, 8737, 8741, 8747, 8753, 8761,
8779, 8783, 8803, 8807, 8819, 8821, 8831,
      8837, 8839, 8849, 8861, 8863, 8867, 8887, 8893, 8923, 8929, 8933, 8941,
8951, 8963, 8969, 8971, 8999, 9001, 9007,
      9011, 9013, 9029, 9041, 9043, 9049, 9059, 9067, 9091, 9103, 9109, 9127,
9133, 9137, 9151, 9157, 9161, 9173, 9181,
      9187, 9199, 9203, 9209, 9221, 9227, 9239, 9241, 9257, 9277, 9281, 9283,
9293, 9311, 9319, 9323, 9337, 9341, 9343,
      9349, 9371, 9377, 9391, 9397, 9403, 9413, 9419, 9421, 9431, 9433, 9437,
9439, 9461, 9463, 9467, 9473, 9479, 9491,
      9497, 9511, 9521, 9533, 9539, 9547, 9551, 9587, 9601, 9613, 9619, 9623,
9629, 9631, 9643, 9649, 9661, 9677, 9679,
      9689, 9697, 9719, 9721, 9733, 9739, 9743, 9749, 9767, 9769, 9781, 9787,
9791, 9803, 9811, 9817, 9829, 9833, 9839,
      9851, 9857, 9859, 9871, 9883, 9887, 9901, 9907, 9923, 9929, 9931, 9941,
9949, 9967, 9973);
   chr64 : Array[1..64] Of char = ('a', 'A', 'b', 'B', 'c', 'C', 'd', 'D', 'e',
'E', 'f', 'F',
      'g', 'G', 'h', 'H', 'i', 'I', 'j', 'J', 'k', 'K', 'l', 'L', 'm', 'M', 'n',
'N', 'o', 'O', 'p',
      'P', 'q', 'Q', 'r', 'R', 's', 'S', 't', 'T', 'u', 'U', 'v', 'V', 'w', 'W',
'x', 'X', 'y', 'Y',
      'z', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '=');
   PGPchr64 : Array[1..64] Of char = ('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
'I', 'J', 'K', 'L', 'M',
      'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b',
'c', 'd', 'e', 'f',
      'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u',
'v', 'w', 'x', 'y',
      'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/');


{$H+}


Procedure zeronetochar8(Var g : char; Const x : String);
Var
   i : Integer;
   b : byte;
Begin
   b := 0;
   For i := 1 To 8 Do
   Begin
      If copy(x, i, 1) = '1' Then
         b := b Or (1 Shl (8 - I));
   End;
   g := chr(b);
End;


Procedure zeronetochar6(Var g : integer; Const x : String);
Var
   I : Integer;
Begin
   G := 0;
   For I := 1 To Length(X) Do
   Begin
      If I > 6 Then
         Break;
      If X[I] <> '0' Then
         G := G Or (1 Shl (6 - I));
   End;
   Inc(G);
End;


Procedure initialize8(Var trans : Array Of String);
```

```
Var
   c1, c2, c3, c4, c5, c6, c7, c8 : integer;
   x : String;
   g : char;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                     For c7 := 0 To 1 Do
                        For c8 := 0 To 1 Do
                        Begin
                           x := '';
                           x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6) + inttostr(c7) + inttostr(c8);
                           zeronetochar8(g, x);
                           trans[ord(g)] := x;
                        End;
End;


Procedure initialize6(Var trans : Array Of String);
Var
   c1, c2, c3, c4, c5, c6 : integer;
   x : String;
   g : integer;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                  Begin
                     x := '';
                     x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6);
                     zeronetochar6(g, x);
                     trans[ord(chr64[g])] := x;
                  End;
End;

Procedure initialize6PGP(Var trans : Array Of String);
Var
   c1, c2, c3, c4, c5, c6 : integer;
   x : String;
   g : integer;
Begin
   For c1 := 0 To 1 Do
      For c2 := 0 To 1 Do
         For c3 := 0 To 1 Do
            For c4 := 0 To 1 Do
               For c5 := 0 To 1 Do
                  For c6 := 0 To 1 Do
                  Begin
                     x := '';
                     x := inttostr(c1) + inttostr(c2) + inttostr(c3) +
inttostr(c4) + inttostr(c5) + inttostr(c6);
                     zeronetochar6(g, x);
                     trans[ord(PGPchr64[g])] := x;
                  End;
End;


// перетворюємо строки довжиною 256 біт у 64 біта

Procedure ConvertBase256to64(Const str256 : String; Var str64 : String);
```

```
Var
   temp : String;
   trans : Array[0..255] Of String;
   i, len6 : longint;
   g : integer;
Begin
   initialize8(trans);
   temp := '';
   For i := 1 To length(str256) Do temp := temp + trans[ord(str256[i])];
   While (length(temp) Mod 6) <> 0 Do temp := temp + '0';
   len6 := length(temp) Div 6;
   str64 := '';
   For i := 1 To len6 Do
   Begin
      zeronetochar6(g, copy(temp, 1, 6));
      str64 := str64 + chr64[g];
      delete(temp, 1, 6);
   End;
End;

// перетворюємо строки довжиною 64 біт у 256 біта

Procedure ConvertBase64to256(Const str64 : String; Var str256 : String);
Var
   temp : String;
   trans : Array[0..255] Of String;
   i, len8 : longint;
   g : char;
Begin
   initialize6(trans);
   temp := '';
   For i := 1 To length(str64) Do temp := temp + trans[ord(str64[i])];
   str256 := '';
   len8 := length(temp) Div 8;
   For i := 1 To len8 Do
   Begin
      zeronetochar8(g, copy(temp, 1, 8));
      str256 := str256 + g;
      delete(temp, 1, 8);
   End;
End;


// перетворюємо строки довжиною 256 біт у 2 біта

Procedure ConvertBase256to2(Const str256 : String; Var str2 : String);
Var
   trans : Array[0..255] Of String;
   i : longint;
Begin
   str2 := '';
   initialize8(trans);
   For i := 1 To length(str256) Do str2 := str2 + trans[ord(str256[i])];
End;

// перетворюємо строки довжиною 64 біт у 2 біта

Procedure ConvertBase64to2(Const str64 : String; Var str2 : String);
Var
   trans : Array[0..255] Of String;
   i : longint;
Begin
   str2 := '';
   initialize6(trans);
   For i := 1 To length(str64) Do str2 := str2 + trans[ord(str64[i])];
End;

// перетворюємо строки довжиною 2 біт у 256 біт
```

```
Procedure ConvertBase2to256(str2 : String; Var str256 : String);
Var
   i, len8 : longint;
   g : char;
Begin
   str256 := '';
   While (length(str2) Mod 8) <> 0 Do str2 := '0' + str2;
   len8 := length(str2) Div 8;
   For i := 1 To len8 Do
   Begin
      zeronetochar8(g, copy(str2, 1, 8));
      str256 := str256 + g;
      delete(str2, 1, 8);
   End;
End;

// перетворюємо строки довжиною 2 біта у 64 біта

Procedure ConvertBase2to64(str2 : String; Var str64 : String);
Var
   i, len6 : longint;
   g : integer;
Begin
   str64 := '';
   While (length(str2) Mod 6) <> 0 Do str2 := '0' + str2;
   len6 := length(str2) Div 6;
   For i := 1 To len6 Do
   Begin
      zeronetochar6(g, copy(str2, 1, 6));
      str64 := str64 + chr64[g];
      delete(str2, 1, 6);
   End;
End;


// перетворюємо строки довжиною 256 біт у 16 біта

Procedure ConvertBase256StringToHexString(Str256 : String; Var HexStr : String);
Var
   i : longint;
   b : byte;
Begin
   HexStr := '';
   For i := 1 To length(str256) Do
   Begin
      b := ord(str256[i]);
      If (b Shr 4) < 10 Then HexStr := HexStr + chr(48 + (b Shr 4))
      Else HexStr := HexStr + chr(55 + (b Shr 4));
      If (b And 15) < 10 Then HexStr := HexStr + chr(48 + (b And 15))
      Else HexStr := HexStr + chr(55 + (b And 15));
   End;
End;



// перетворюємо строки довжиною 16 біт у 256 біт

Procedure ConvertHexStringToBase256String(HexStr : String; Var Str256 : String);
Var
   i : longint;
   b, h1, h2 : byte;
Begin
   Str256 := '';
   For i := 1 To (length(Hexstr) Div 2) Do
   Begin
      h2 := ord(HexStr[2 * i]);
      h1 := ord(HexStr[2 * i - 1]);
      If h1 < 58 Then b := ((h1 - 48) Shl 4) Else b := ((h1 - 55) Shl 4);
```

```
      If h2 < 58 Then b := (b Or (h2 - 48)) Else b := (b Or (h2 - 55));
      Str256 := Str256 + chr(b);
   End;
End;


// перетворюємо строки довжиною 256 біт у 64 біта для PGP

Procedure PGPConvertBase256to64(Var str256, str64 : String);
Var
   temp, x, a : String;
   i, len6 : longint;
   g : integer;
   trans : Array[0..255] Of String;
Begin
   initialize8(trans);
   temp := '';
   For i := 1 To length(str256) Do temp := temp + trans[ord(str256[i])];
   If (length(temp) Mod 6) = 0 Then a := '' Else
      If (length(temp) Mod 6) = 4 Then
      Begin
         temp := temp + '00';
         a := '='
      End
      Else
      Begin
         temp := temp + '0000';
         a := '=='
      End;
   str64 := '';
   len6 := length(temp) Div 6;
   For i := 1 To len6 Do
   Begin
      x := copy(temp, 1, 6);
      zeronetochar6(g, x);
      str64 := str64 + PGPchr64[g];
      delete(temp, 1, 6);
   End;
   str64 := str64 + a;
End;

// перетворюємо строки довжиною 64 біт у 256 біт для PGP

Procedure PGPConvertBase64to256(str64 : String; Var str256 : String);
Var
   temp, x : String;
   i, j, len8 : longint;
   g : char;
   trans : Array[0..255] Of String;
Begin
   initialize6PGP(trans);
   temp := '';
   str256 := '';
   If str64[length(str64) - 1] = '=' Then j := 2 Else
      If str64[length(str64)] = '=' Then j := 1 Else j := 0;
   For i := 1 To (length(str64) - j) Do temp := temp + trans[ord(str64[i])];
   If j <> 0 Then delete(temp, length(temp) - 2 * j + 1, 2 * j);
   len8 := length(temp) Div 8;
   For i := 1 To len8 Do
   Begin
      x := copy(temp, 1, 8);
      zeronetochar8(g, x);
      str256 := str256 + g;
      delete(temp, 1, 8);
   End;
End;

// перетворюємо строки довжиною 64 біт у 2 біта для PGP
```

```
Procedure PGPConvertBase64to2(str64 : String; Var str2 : String);
Var
   i, j : longint;
   trans : Array[0..255] Of String;
Begin
   str2 := '';
   initialize6(trans);
   If str64[length(str64) - 1] = '=' Then j := 2 Else
      If str64[length(str64)] = '=' Then j := 1 Else j := 0;
   For i := 1 To (length(str64) - j) Do str2 := str2 + trans[ord(str64[i])];
   delete(str2, length(str2) - 2 * j + 1, 2 * j);
End;


// перетворюємо строки довжиною 10 біт у  FGInt

Procedure Base10StringToFGInt(Base10 : String; Var FGInt : TFGInt);
Var
   i, size : longint;
   j : int64;
   S : String;
   sign : TSign;

   Procedure GIntDivByIntBis1(Var GInt : TFGInt; by : int64; Var modres :
int64);
   Var
      i, size : longint;
      rest : int64;
   Begin
      size := GInt.Number[0];
      modres := 0;
      For i := size Downto 1 Do
      Begin
         modres := modres * 1000000000;
         rest := modres + GInt.Number[i];
         GInt.Number[i] := rest Div by;
         modres := rest Mod by;
      End;
      While (GInt.Number[size] = 0) And (size > 1) Do size := size - 1;
      If size <> GInt.Number[0] Then
      Begin
         SetLength(GInt.Number, size + 1);
         GInt.Number[0] := size;
      End;
   End;

Begin
   While (Not (Base10[1] In ['-', '0'..'9'])) And (length(Base10) > 1) Do
      delete(Base10, 1, 1);
   If copy(Base10, 1, 1) = '-' Then
   Begin
      Sign := negative;
      delete(Base10, 1, 1);
   End
   Else Sign := positive;
   While (length(Base10) > 1) And (copy(Base10, 1, 1) = '0') Do delete(Base10,
1, 1);
   size := length(Base10) Div 9;
   If (length(Base10) Mod 9) <> 0 Then size := size + 1;
   SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
   For i := 1 To size - 1 Do
   Begin
      FGInt.Number[i] := StrToInt(copy(Base10, length(Base10) - 8, 9));
      delete(Base10, length(Base10) - 8, 9);
   End;
   FGInt.Number[size] := StrToInt(Base10);

   S := '';
```

```
   While (FGInt.Number[0] <> 1) Or (FGInt.Number[1] <> 0) Do
   Begin
      GIntDivByIntBis1(FGInt, 2, j);
      S := inttostr(j) + S;
   End;
   S := '0' + S;
   FGIntDestroy(FGInt);
   Base2StringToFGInt(S, FGInt);
   FGInt.Sign := sign;
End;


// перетворюємо  FGInt в рядок 10 біт

Procedure FGIntToBase10String(Const FGInt : TFGInt; Var Base10 : String);
Var
   S : String;
   j : int64;
   temp : TFGInt;
Begin
   FGIntCopy(FGInt, temp);
   Base10 := '';
   While (temp.Number[0] > 1) Or (temp.Number[1] > 0) Do
   Begin
      FGIntDivByIntBis(temp, 1000000000, j);
      S := IntToStr(j);
      While Length(S) < 9 Do S := '0' + S;
      Base10 := S + Base10;
   End;
   Base10 := '0' + Base10;
   While (length(Base10) > 1) And (Base10[1] = '0') Do delete(Base10, 1, 1);
   If FGInt.Sign = negative Then Base10 := '-' + Base10;
End;



// Видаляємо FGInt для звільнення пам'яті

Procedure FGIntDestroy(Var FGInt : TFGInt);
Begin
   FGInt.Number := Nil;
End;



Function FGIntCompareAbs(Const FGInt1, FGInt2 : TFGInt) : TCompare;
Var
   size1, size2, i : longint;
Begin
   FGIntCompareAbs := Er;
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   If size1 > size2 Then FGIntCompareAbs := Lt Else
      If size1 < size2 Then FGIntCompareAbs := St Else
      Begin
         i := size2;
         While (FGInt1.Number[i] = FGInt2.Number[i]) And (i > 1) Do i := i - 1;
         If FGInt1.Number[i] = FGInt2.Number[i] Then FGIntCompareAbs := Eq Else
            If FGInt1.Number[i] < FGInt2.Number[i] Then FGIntCompareAbs := St
Else
               If FGInt1.Number[i] > FGInt2.Number[i] Then FGIntCompareAbs :=
Lt;
      End;
End;


// Додаємо 2 FGInts, FGInt1 + FGInt2 = Sum

Procedure FGIntAdd(Const FGInt1, FGInt2 : TFGInt; Var Sum : TFGInt);
```

```
Var
   i, size1, size2, size : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   If size1 < size2 Then FGIntAdd(FGInt2, FGInt1, Sum) Else
   Begin
      If FGInt1.Sign = FGInt2.Sign Then
      Begin
         Sum.Sign := FGInt1.Sign;
         SetLength(Sum.Number, size1 + 2);
         rest := 0;
         For i := 1 To size2 Do
         Begin
            Trest := FGInt1.Number[i] + FGInt2.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         For i := (size2 + 1) To size1 Do
         Begin
            Trest := FGInt1.Number[i] + rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         size := size1 + 1;
         Sum.Number[0] := size;
         Sum.Number[size] := rest;
         While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
         If Sum.Number[0] > size Then SetLength(Sum.Number, size + 1);
         Sum.Number[0] := size;
      End
      Else
      Begin
         If FGIntCompareAbs(FGInt2, FGInt1) = Lt Then FGIntAdd(FGInt2, FGInt1,
Sum)
         Else
         Begin
            SetLength(Sum.Number, size1 + 1);
            rest := 0;
            For i := 1 To size2 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] - FGInt2.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            For i := (size2 + 1) To size1 Do
            Begin
               Trest := 2147483648 + FGInt1.Number[i] + rest;
               Sum.Number[i] := Trest And 2147483647;
               If (Trest > 2147483647) Then rest := 0 Else rest := -1;
            End;
            size := size1;
            While (Sum.Number[size] = 0) And (size > 1) Do size := size - 1;
            If size < size1 Then
            Begin
               SetLength(Sum.Number, size + 1);
            End;
            Sum.Number[0] := size;
            Sum.Sign := FGInt1.Sign;
         End;
      End;
   End;
End;


Procedure FGIntChangeSign(Var FGInt : TFGInt);
Begin
```

```
   If FGInt.Sign = negative Then FGInt.Sign := positive Else FGInt.Sign :=
negative;
End;


// Віднімаємо 2 FGInts, FGInt1 - FGInt2 = dif

Procedure FGIntSub(Var FGInt1, FGInt2, dif : TFGInt);
Begin
   FGIntChangeSign(FGInt2);
   FGIntAdd(FGInt1, FGInt2, dif);
   FGIntChangeSign(FGInt2);
End;


// Перемножуємоа FGInt з цілим, FGInt * by = res, by < 1000000000

Procedure FGIntMulByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64);
Var
   i, size : longint;
   Trest, rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(res.Number, size + 2);
   rest := 0;
   For i := 1 To size Do
   Begin
      Trest := FGInt.Number[i] * by + rest;
      res.Number[i] := Trest And 2147483647;
      rest := Trest Shr 31;
   End;
   If rest <> 0 Then
   Begin
      size := size + 1;
      Res.Number[size] := rest;
   End
   Else SetLength(Res.Number, size + 1);
   Res.Number[0] := size;
   Res.Sign := FGInt.Sign;
End;


// перемножуємо FGInt з цілим, FGInt * by = res, by < 1000000000

Procedure FGIntMulByIntbis(Var FGInt : TFGInt; by : int64);
Var
   i, size : longint;
   Trest, rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(FGInt.Number, size + 2);
   rest := 0;
   For i := 1 To size Do
   Begin
      Trest := FGInt.Number[i] * by + rest;
      FGInt.Number[i] := Trest And 2147483647;
      rest := Trest Shr 31;
   End;
   If rest <> 0 Then
   Begin
      size := size + 1;
      FGInt.Number[size] := rest;
   End
   Else SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
End;


// ділимо FGInt на ціле, FGInt = res * by + modres
```

```
Procedure FGIntDivByInt(Const FGInt : TFGInt; Var res : TFGInt; by : int64; Var
modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   SetLength(res.Number, size + 1);
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres Or FGInt.Number[i];
      res.Number[i] := rest Div by;
      modres := rest Mod by;
   End;
   While (res.Number[size] = 0) And (size > 1) Do size := size - 1;
   SetLength(res.Number, size + 1);
   res.Number[0] := size;
   Res.Sign := FGInt.Sign;
End;


// ділимо FGInt на ціле, FGInt = FGInt * by + modres

Procedure FGIntDivByIntBis(Var FGInt : TFGInt; by : int64; Var modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres Or FGInt.Number[i];
      FGInt.Number[i] := rest Div by;
      modres := rest Mod by;
   End;
   While (FGInt.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size <> FGInt.Number[0] Then
   Begin
      SetLength(FGInt.Number, size + 1);
      FGInt.Number[0] := size;
   End;
End;


// Беремо FGInt по модулю цілого числа, FGInt mod by = modres

Procedure FGIntModByInt(Const FGInt : TFGInt; by : int64; Var modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := FGInt.Number[0];
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres + FGInt.Number[i];
      modres := rest Mod by;
   End;
End;
```

```
// повертаємо FGInt по модулю

Procedure FGIntAbs(Var FGInt : TFGInt);
Begin
   FGInt.Sign := positive;
End;

// Корiюємо FGInt1 в FGInt2

Procedure FGIntCopy(Const FGInt1 : TFGInt; Var FGInt2 : TFGInt);
Begin
   FGInt2.Sign := FGInt1.Sign;
   FGInt2.Number := Nil;
   FGInt2.Number := Copy(FGInt1.Number, 0, FGInt1.Number[0] + 1);
End;

// Зрушуємо FGInt улiво на 2 бiта FGInt = FGInt * 2

Procedure FGIntShiftLeft(Var FGInt : TFGInt);
Var
   l, m : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   l := 0;
   For i := 1 To Size Do
   Begin
      m := FGInt.Number[i] Shr 30;
      FGInt.Number[i] := ((FGInt.Number[i] Shl 1) Or l) And 2147483647;
      l := m;
   End;
   If l <> 0 Then
   Begin
      setlength(FGInt.Number, size + 2);
      FGInt.Number[size + 1] := 1;
      FGInt.Number[0] := size + 1;
   End;
End;

// Зрушуємо FGInt вправо на 2 бiта, FGInt = FGInt div 2

Procedure FGIntShiftRight(Var FGInt : TFGInt);
Var
   l, m : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   l := 0;
   For i := size Downto 1 Do
   Begin
      m := FGInt.Number[i] And 1;
      FGInt.Number[i] := (FGInt.Number[i] Shr 1) Or l;
      l := m Shl 30;
   End;
   If (FGInt.Number[size] = 0) And (size > 1) Then
   Begin
      setlength(FGInt.Number, size);
      FGInt.Number[0] := size - 1;
   End;
End;


// FGInt = FGInt / 2147483648

Procedure FGIntShiftRightBy31(Var FGInt : TFGInt);
Var
   size : longint;
Begin
   size := FGInt.Number[0];
```

```
   If size > 1 Then
   Begin
      FGInt.Number := Copy(FGInt.Number, 1, Size);
      FGInt.Number[0] := size - 1;
   End
   Else FGInt.Number[1] := 0;
End;


// FGInt1 = FGInt1 + FGInt2, FGInt1 > FGInt2

Procedure FGIntAddBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := FGInt1.Number[i] + FGInt2.Number[i] + rest;
      rest := Trest Shr 31;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := FGInt1.Number[i] + rest;
      rest := Trest Shr 31;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   If rest <> 0 Then
   Begin
      SetLength(FGInt1.Number, size1 + 2);
      FGInt1.Number[0] := size1 + 1;
      FGInt1.Number[size1 + 1] := rest;
   End;
End;


// FGInt1 = FGInt1 - FGInt2, використовується тільки якщо 0 < FGInt2 < FGInt1

Procedure FGIntSubBis(Var FGInt1 : TFGInt; Const FGInt2 : TFGInt);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := 2147483648 + FGInt1.Number[i] - FGInt2.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := 2147483648 + FGInt1.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      FGInt1.Number[i] := Trest And 2147483647;
   End;
   i := size1;
   While (FGInt1.Number[i] = 0) And (i > 1) Do i := i - 1;
   If i < size1 Then
   Begin
      SetLength(FGInt1.Number, i + 1);
```

```
      FGInt1.Number[0] := i;
   End;
End;


// Перемножуємо 2 FGInts, FGInt1 * FGInt2 = Prod

Procedure FGIntMul(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt);
Var
   i, j, size, size1, size2 : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   size := size1 + size2;
   SetLength(Prod.Number, size + 1);
   For i := 1 To size Do Prod.Number[i] := 0;

   For i := 1 To size2 Do
   Begin
      rest := 0;
      For j := 1 To size1 Do
      Begin
         Trest := Prod.Number[j + i - 1] + FGInt1.Number[j] * FGInt2.Number[i] +
rest;
         Prod.Number[j + i - 1] := Trest And 2147483647;
         rest := Trest Shr 31;
      End;
      Prod.Number[i + size1] := rest;
   End;

   Prod.Number[0] := size;
   While (Prod.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size < Prod.Number[0] Then
   Begin
      SetLength(Prod.Number, size + 1);
      Prod.Number[0] := size;
   End;
   If FGInt1.Sign = FGInt2.Sign Then Prod.Sign := Positive Else prod.Sign :=
negative;
End;


// Підводимо в квадрат FGInt, FGIntI = Square

Procedure FGIntSquare(Const FGInt : TFGInt; Var Square : TFGInt);
Var
   size, size1, i, j : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt.Number[0];
   size := 2 * size1;
   SetLength(Square.Number, size + 1);
   Square.Number[0] := size;
   For i := 1 To size Do Square.Number[i] := 0;
   For i := 1 To size1 Do
   Begin
      Trest := Square.Number[2 * i - 1] + FGInt.Number[i] * FGInt.Number[i];
      Square.Number[2 * i - 1] := Trest And 2147483647;
      rest := Trest Shr 31;
      For j := i + 1 To size1 Do
      Begin
         Trest := Square.Number[i + j - 1] + 2 * FGInt.Number[i] *
FGInt.Number[j] + rest;
         Square.Number[i + j - 1] := Trest And 2147483647;
         rest := Trest Shr 31;
      End;
      Square.Number[i + size1] := rest;
   End;
```

```
   Square.Sign := positive;
   While (Square.Number[size] = 0) And (size > 1) Do size := size - 1;
   If size < 2 * size1 Then
   Begin
      SetLength(Square.Number, size + 1);
      Square.Number[0] := size;
   End;
End;


// Перетворюємо FGInt у бінарну рядок

Procedure FGIntToBase2String(Const FGInt : TFGInt; Var S : String);
Var
   i : longint;
   j : integer;
Begin
   S := '';
   For i := 1 To FGInt.Number[0] Do
   Begin
      For j := 0 To 30 Do S := inttostr(1 And (FGInt.Number[i] Shr j)) + S;
   End;
   While (length(S) > 1) And (S[1] = '0') Do delete(S, 1, 1);
   If S = '' Then S := '0';
End;


Procedure Base2StringToFGInt(S : String; Var FGInt : TFGInt);
Var
   i, j, size : longint;
Begin
   while (S[1]='0') and (length(S)>1) do delete(S,1,1);
   size := length(S) Div 31;
   If (length(S) Mod 31) <> 0 Then size := size + 1;
   SetLength(FGInt.Number, size + 1);
   FGInt.Number[0] := size;
   j := 1;
   FGInt.Number[j] := 0;
   i := 0;
   While length(S) > 0 Do
   Begin
      If S[length(S)] = '1' Then
         FGInt.Number[j] := FGInt.Number[j] Or (1 Shl i);
      i := i + 1;
      If i = 31 Then
      Begin
         i := 0;
         j := j + 1;
         If j <= size Then FGInt.Number[j] := 0;
      End;
      delete(S, length(S), 1);
   End;
   FGInt.Sign := positive;
End;


// перетворюємо FGInt у рядок 256 біт

Procedure FGIntToBase256String(Const FGInt : TFGInt; Var str256 : String);
Var
   temp1 : String;
   i, len8 : longint;
   g : char;
Begin
   FGIntToBase2String(FGInt, temp1);
   While (length(temp1) Mod 8) <> 0 Do temp1 := '0' + temp1;
   len8 := length(temp1) Div 8;
   str256 := '';
   For i := 1 To len8 Do
```

```
   Begin
      zeronetochar8(g, copy(temp1, 1, 8));
      str256 := str256 + g;
      delete(temp1, 1, 8);
   End;
End;


Procedure Base256StringToFGInt(str256 : String; Var FGInt : TFGInt);
Var
   temp1 : String;
   i : longint;
   trans : Array[0..255] Of String;
Begin
   temp1 := '';
   initialize8(trans);
   For i := 1 To length(str256) Do temp1 := temp1 + trans[ord(str256[i])];
   While (temp1[1] = '0') And (temp1 <> '0') Do delete(temp1, 1, 1);
   Base2StringToFGInt(temp1, FGInt);
End;

// Перетворюємо MPI (Multiple Precision Integer, для PGP) у FGInt

Procedure PGPMPIToFGInt(PGPMPI : String; Var FGInt : TFGInt);
Var
   temp : String;
Begin
   temp := PGPMPI;
   delete(temp, 1, 2);
   Base256StringToFGInt(temp, FGInt);
End;


Procedure FGIntToPGPMPI(FGInt : TFGInt; Var PGPMPI : String);
Var
   len, i : word;
   c : char;
   b : byte;
Begin
   FGIntToBase256String(FGInt, PGPMPI);
   len := length(PGPMPI) * 8;
   c := PGPMPI[1];
   For i := 7 Downto 0 Do If (ord(c) Shr i) = 0 Then len := len - 1 Else break;
   b := len Mod 256;
   PGPMPI := chr(b) + PGPMPI;
   b := len Div 256;
   PGPMPI := chr(b) + PGPMPI;
End;



// Піднімаємо у ступінь FGInt, FGInt^exp = res

Procedure FGIntExp(Const FGInt, exp : TFGInt; Var res : TFGInt);
Var
   temp2, temp3 : TFGInt;
   S : String;
   i : longint;
Begin
   FGIntToBase2String(exp, S);
   If S[length(S)] = '0' Then Base10StringToFGInt('1', res) Else
FGIntCopy(FGInt, res);
   FGIntCopy(FGInt, temp2);
   If length(S) > 1 Then
      For i := (length(S) - 1) Downto 1 Do
      Begin
         FGIntSquare(temp2, temp3);
         FGIntCopy(temp3, temp2);
```

```
         If S[i] = '1' Then
         Begin
            FGIntMul(res, temp2, temp3);
            FGIntCopy(temp3, res);
         End;
      End;
End;


// Розраховуємо FGInt! = FGInt * (FGInt - 1) * (FGInt - 2) * ... * 3 * 2 * 1

Procedure FGIntFac(Const FGInt : TFGInt; Var res : TFGInt);
Var
   one, temp, temp1 : TFGInt;
Begin
   FGIntCopy(FGInt, temp);
   Base10StringToFGInt('1', res);
   Base10StringToFGInt('1', one);

   While Not (FGIntCompareAbs(temp, one) = Eq) Do
   Begin
      FGIntMul(temp, res, temp1);
      FGIntCopy(temp1, res);
      FGIntSubBis(temp, one);
   End;

   FGIntDestroy(one);
   FGIntDestroy(temp);
End;


// FGInt = FGInt * 2147483648

Procedure FGIntShiftLeftBy31(Var FGInt : TFGInt);
Var
   f1, f2 : int64;
   i, size : longint;
Begin
   size := FGInt.Number[0];
   SetLength(FGInt.Number, size + 2);
   f1 := 0;
   For i := 1 To (size + 1) Do
   Begin
      f2 := FGInt.Number[i];
      FGInt.Number[i] := f1;
      f1 := f2;
   End;
   FGInt.Number[0] := size + 1;
End;


// Ділимо 2 FGInts, FGInt1 = FGInt2 * QFGInt + MFGInt, MFGInt позитивне

Procedure FGIntDivMod(Var FGInt1, FGInt2, QFGInt, MFGInt : TFGInt);
Var
   one, zero, temp1, temp2 : TFGInt;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
```

```
      setlength(QFGInt.Number, s + 2);
      QFGInt.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         FGIntShiftLeftBy31(temp1);
         QFGInt.Number[t] := 0;
      End;
      j := s + 1;
      QFGInt.Number[j] := 0;
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
               QFGInt.Number[j] := QFGInt.Number[j] + i;
               If FGIntCompareAbs(MFGInt, temp2) <> St Then
               Begin
                  QFGInt.Number[j] := QFGInt.Number[j] + i;
                  FGIntSubBis(MFGInt, temp2);
               End;
            End Else
            Begin
               QFGInt.Number[j] := QFGInt.Number[j] + 1;
               FGIntSubBis(MFGInt, temp1);
            End;
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
            Begin
               FGIntShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
   Else Base10StringToFGInt('0', QFGInt);
   s := QFGInt.Number[0];
   While (s > 1) And (QFGInt.Number[s] = 0) Do s := s - 1;
   If s < QFGInt.Number[0] Then
   Begin
      setlength(QFGInt.Number, s + 1);
      QFGInt.Number[0] := s;
   End;
   QFGInt.Sign := positive;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntadd(QFGInt, one, temp1);
         FGIntCopy(temp1, QFGInt);
         FGIntDestroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
```

```
   FGIntDestroy(one);
   FGIntDestroy(zero);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// Розраховуємо MFGInt

Procedure FGIntDiv(Var FGInt1, FGInt2, QFGInt : TFGInt);
Var
   one, zero, temp1, temp2, MFGInt : TFGInt;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
      setlength(QFGInt.Number, s + 2);
      QFGInt.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         FGIntShiftLeftBy31(temp1);
         QFGInt.Number[t] := 0;
      End;
      j := s + 1;
      QFGInt.Number[j] := 0;
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
               QFGInt.Number[j] := QFGInt.Number[j] + i;
               If FGIntCompareAbs(MFGInt, temp2) <> St Then
               Begin
                  QFGInt.Number[j] := QFGInt.Number[j] + i;
                  FGIntSubBis(MFGInt, temp2);
               End;
            End Else
            Begin
               QFGInt.Number[j] := QFGInt.Number[j] + 1;
               FGIntSubBis(MFGInt, temp1);
            End;
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
            Begin
               FGIntShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
```

```
   Else Base10StringToFGInt('0', QFGInt);
   s := QFGInt.Number[0];
   While (s > 1) And (QFGInt.Number[s] = 0) Do s := s - 1;
   If s < QFGInt.Number[0] Then
   Begin
      setlength(QFGInt.Number, s + 1);
      QFGInt.Number[0] := s;
   End;
   QFGInt.Sign := positive;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntadd(QFGInt, one, temp1);
         FGIntCopy(temp1, QFGInt);
         FGIntDestroy(temp1);
         FGIntsub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
      If s2 = positive Then QFGInt.Sign := negative;
   End
   Else QFGInt.Sign := s2;
   FGIntDestroy(one);
   FGIntDestroy(zero);
   FGIntDestroy(MFGInt);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// MFGInt = FGInt1 mod FGInt2

Procedure FGIntMod(Var FGInt1, FGInt2, MFGInt : TFGInt);
Var
   one, zero, temp1, temp2 : TFGInt;
   s1, s2 : TSign;
   i : int64;
   s, t : longint;
Begin
   s1 := FGInt1.Sign;
   s2 := FGInt2.Sign;
   FGIntAbs(FGInt1);
   FGIntAbs(FGInt2);
   FGIntCopy(FGInt1, MFGInt);
   FGIntCopy(FGInt2, temp1);

   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      s := FGInt1.Number[0] - FGInt2.Number[0];
      For t := 1 To s Do FGIntShiftLeftBy31(temp1);
      While FGIntCompareAbs(MFGInt, FGInt2) <> St Do
      Begin
         While FGIntCompareAbs(MFGInt, temp1) <> St Do
         Begin
            If MFGInt.Number[0] > temp1.Number[0] Then
               i := (2147483648 * MFGInt.Number[MFGInt.Number[0]] +
MFGInt.Number[MFGInt.Number[0] - 1]) Div (temp1.Number[temp1.Number[0]] + 1)
            Else i := MFGInt.Number[MFGInt.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               FGIntCopy(temp1, temp2);
               FGIntMulByIntBis(temp2, i);
               FGIntSubBis(MFGInt, temp2);
```

```
               If FGIntCompareAbs(MFGInt, temp2) <> St Then FGIntSubBis(MFGInt,
temp2);
            End Else FGIntSubBis(MFGInt, temp1);
//         If FGIntCompareAbs(MFGInt, temp1) <> St Then
FGIntSubBis(MFGInt,temp1);
         End;
         If MFGInt.Number[0] <= temp1.Number[0] Then
            If FGIntCompareAbs(temp1, FGInt2) <> Eq Then
FGIntShiftRightBy31(temp1);
      End;
   End;

   FGIntDestroy(temp1);
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   If s1 = negative Then
   Begin
      If FGIntCompareAbs(MFGInt, zero) <> Eq Then
      Begin
         FGIntSub(FGInt2, MFGInt, temp1);
         FGIntCopy(temp1, MFGInt);
      End;
   End;
   FGIntDestroy(one);
   FGIntDestroy(zero);

   FGInt1.Sign := s1;
   FGInt2.Sign := s2;
End;


// підводимо у квадрат FGInt за модулем  Modb, FGInt^2 mod Modb = FGIntSM

Procedure FGIntSquareMod(Var FGInt, Modb, FGIntSM : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntSquare(FGInt, temp);
   FGIntMod(temp, Modb, FGIntSM);
   FGIntDestroy(temp);
End;


// Додаємо 2 FGInts за модулем, (FGInt1 + FGInt2) mod base = FGIntres

Procedure FGIntAddMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntadd(FGInt1, FGInt2, temp);
   FGIntMod(temp, base, FGIntres);
   FGIntDestroy(temp);
End;


// Перемножуємо 2 FGInts за модулем, (FGInt1 * FGInt2) mod base = FGIntres

Procedure FGIntMulMod(Var FGInt1, FGInt2, base, FGIntres : TFGInt);
Var
   temp : TFGInt;
Begin
   FGIntMul(FGInt1, FGInt2, temp);
   FGIntMod(temp, base, FGIntres);
   FGIntDestroy(temp);
End;


// Підводимо у ступінь 2 FGInts за модулем, (FGInt1 ^ FGInt2) mod modb = res
```

```
Procedure FGIntModExp(Var FGInt, exp, modb, res : TFGInt);
Var
   temp2, temp3 : TFGInt;
   i : longint;
   S : String;
Begin
   If (Modb.Number[1] Mod 2) = 1 Then
   Begin
      FGIntMontgomeryModExp(FGInt, exp, modb, res);
      exit;
   End;
   FGIntToBase2String(exp, S);
   Base10StringToFGInt('1', res);
   FGIntcopy(FGInt, temp2);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         FGIntmulMod(res, temp2, modb, temp3);
         FGIntCopy(temp3, res);
      End;
      FGIntSquareMod(temp2, Modb, temp3);
      FGIntCopy(temp3, temp2);
   End;
   FGIntDestroy(temp2);
End;


// Процедура підводення у ступень за Монтгомері

Procedure FGIntModBis(Const FGInt : TFGInt; Var FGIntOut : TFGInt; b : longint;
head : int64);
Var
   i : longint;
Begin
   If b <= FGInt.Number[0] Then
   Begin
      FGIntOut.Number := Copy(FGInt.Number, 0, b + 1);
      FGIntOut.Number[b] := FGIntOut.Number[b] And head;
      i := b;
      While (FGIntOut.Number[i] = 0) And (i > 1) Do i := i - 1;
      If i < b Then SetLength(FGIntOut.Number, i + 1);
      FGIntOut.Number[0] := i;
      FGIntOut.Sign := positive;
   End Else FGIntCopy(FGInt, FGIntOut);
End;


Procedure FGIntMulModBis(Const FGInt1, FGInt2 : TFGInt; Var Prod : TFGInt; b :
longint; head : int64);
Var
   i, j, size, size1, size2, t : longint;
   rest, Trest : int64;
Begin
   size1 := FGInt1.Number[0];
   size2 := FGInt2.Number[0];
   size := min(b, size1 + size2);
   SetLength(Prod.Number, size + 1);
   For i := 1 To size Do Prod.Number[i] := 0;

   For i := 1 To size2 Do
   Begin
      rest := 0;
      t := min(size1, b - i + 1);
      For j := 1 To t Do
      Begin
         Trest := Prod.Number[j + i - 1] + FGInt1.Number[j] * FGInt2.Number[i] +
rest;
```

```
            Prod.Number[j + i - 1] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         If (i + size1) <= b Then Prod.Number[i + size1] := rest;
      End;

      Prod.Number[0] := size;
      If size = b Then Prod.Number[b] := Prod.Number[b] And head;
      While (Prod.Number[size] = 0) And (size > 1) Do size := size - 1;
      If size < Prod.Number[0] Then
      Begin
         SetLength(Prod.Number, size + 1);
         Prod.Number[0] := size;
      End;
      If FGInt1.Sign = FGInt2.Sign Then Prod.Sign := Positive Else prod.Sign :=
negative;
End;


Procedure FGIntMontgomeryMod(Const GInt, base, baseInv : TFGInt; Var MGInt :
TFGInt; b : longint; head : int64);
Var
   m, temp, temp1 : TFGInt;
   r : int64;
Begin
   FGIntModBis(GInt, temp, b, head);
   FGIntMulModBis(temp, baseInv, m, b, head);
   FGIntMul(m, base, temp1);
   FGIntDestroy(temp);
   FGIntAdd(temp1, GInt, temp);
   FGIntDestroy(temp1);
   MGInt.Number := copy(temp.Number, b - 1, temp.Number[0] - b + 2);
   MGInt.Sign := positive;
   MGInt.Number[0] := temp.Number[0] - b + 1;
   FGIntDestroy(temp);
   If (head Shr 30) = 0 Then FGIntDivByIntBis(MGInt, head + 1, r)
   Else FGIntShiftRightBy31(MGInt);
   If FGIntCompareAbs(MGInt, base) <> St Then FGIntSubBis(MGInt, base);
   FGIntDestroy(temp);
   FGIntDestroy(m);
End;


Procedure FGIntMontgomeryModExp(Var FGInt, exp, modb, res : TFGInt);
Var
   temp2, temp3, baseInv, r : TFGInt;
   i, j, t, b : longint;
   S : String;
   head : int64;
Begin
   FGIntToBase2String(exp, S);
   t := modb.Number[0];
   b := t;

   If (modb.Number[t] Shr 30) = 1 Then t := t + 1;
   setlength(r.Number, t + 1);
   r.Number[0] := t;
   r.Sign := positive;
   For i := 1 To t Do r.Number[i] := 0;
   If t = modb.Number[0] Then
   Begin
      head := 2147483647;
      For j := 29 Downto 0 Do
      Begin
         head := head Shr 1;
         If (modb.Number[t] Shr j) = 1 Then
         Begin
            r.Number[t] := 1 Shl (j + 1);
            break;
```

```
         End;
      End;
   End
   Else
   Begin
      r.Number[t] := 1;
      head := 2147483647;
   End;

   FGIntModInv(modb, r, temp2);
   If temp2.Sign = negative Then FGIntCopy(temp2, BaseInv)
   Else
   Begin
      FGIntCopy(r, BaseInv);
      FGIntSubBis(BaseInv, temp2);
   End;
//   FGIntBezoutBachet(r, modb, temp2, BaseInv);
   FGIntAbs(BaseInv);
   FGIntDestroy(temp2);
   FGIntMod(r, modb, res);
   FGIntMulMod(FGInt, res, modb, temp2);
   FGIntDestroy(r);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         FGIntmul(res, temp2, temp3);
         FGIntDestroy(res);
         FGIntMontgomeryMod(temp3, modb, baseinv, res, b, head);
         FGIntDestroy(temp3);
      End;
      FGIntSquare(temp2, temp3);
      FGIntDestroy(temp2);
      FGIntMontgomeryMod(temp3, modb, baseinv, temp2, b, head);
      FGIntDestroy(temp3);
   End;
   FGIntDestroy(temp2);
   FGIntMontgomeryMod(res, modb, baseinv, temp3, b, head);
   FGIntCopy(temp3, res);
   FGIntDestroy(temp3);
   FGIntDestroy(baseinv);
End;


// розраховуємо найбільший загальний дільник 2 FGInts

Procedure FGIntGCD(Const FGInt1, FGInt2 : TFGInt; Var GCD : TFGInt);
Var
   k : TCompare;
   zero, temp1, temp2, temp3 : TFGInt;
Begin
   k := FGIntCompareAbs(FGInt1, FGInt2);
   If (k = Eq) Then FGIntCopy(FGInt1, GCD) Else
      If (k = St) Then FGIntGCD(FGInt2, FGInt1, GCD) Else
      Begin
         Base10StringToFGInt('0', zero);
         FGIntCopy(FGInt1, temp1);
         FGIntCopy(FGInt2, temp2);
         While (temp2.Number[0] <> 1) Or (temp2.Number[1] <> 0) Do
         Begin
            FGIntMod(temp1, temp2, temp3);
            FGIntCopy(temp2, temp1);
            FGIntCopy(temp3, temp2);
            FGIntDestroy(temp3);
         End;
         FGIntCopy(temp1, GCD);
         FGIntDestroy(temp2);
         FGIntDestroy(zero);
```

```
      End;
End;


// розраховуємо найменше загальне кратне 2 FGInts

Procedure FGIntLCM(Const FGInt1, FGInt2 : TFGInt; Var LCM : TFGInt);
Var
   temp1, temp2 : TFGInt;
Begin
   FGIntGCD(FGInt1, FGInt2, temp1);
   FGIntmul(FGInt1, FGInt2, temp2);
   FGIntdiv(temp2, temp1, LCM);
   FGIntDestroy(temp1);
   FGIntDestroy(temp2);
End;


// Знаходження взаємо простого FGInt до 9999 та зупинка коли знайдено таке
число, повертає ok=false

Procedure FGIntTrialDiv9999(Const FGInt : TFGInt; Var ok : boolean);
Var
   j : int64;
   i : integer;
Begin
   If ((FGInt.Number[1] Mod 2) = 0) Then ok := false
   Else
   Begin
      i := 0;
      ok := true;
      While ok And (i < 1228) Do
      Begin
         i := i + 1;
         FGIntmodbyint(FGInt, primes[i], j);
         If j = 0 Then ok := false;
      End;
   End;
End;
// Генератор випадкових чисел

Procedure FGIntRandom1(Var Seed, RandomFGInt : TFGInt);
Var
   temp, base : TFGInt;
Begin
   Base10StringToFGInt('281474976710656', base);
   Base10StringToFGInt('44485709377909', temp);
   FGIntMulMod(seed, temp, base, RandomFGInt);
   FGIntDestroy(temp);
   FGIntDestroy(base);
End;


// Тест на простоту числа FGIntp методом Рабіна-Мілера, повертає ok=true якщо
FGIntp пройшло тест

Procedure FGIntRabinMiller(Var FGIntp : TFGInt; nrtest : integer; Var ok :
boolean);
Var
   j, b, i : int64;
   m, z, temp1, temp2, temp3, zero, one, two, pmin1 : TFGInt;
   ok1, ok2 : boolean;
Begin
   randomize;
   j := 0;
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   Base10StringToFGInt('2', two);
   FGIntsub(FGIntp, one, temp1);
```

```
   FGIntsub(FGIntp, one, pmin1);

   b := 0;
   While (temp1.Number[1] Mod 2) = 0 Do
   Begin
      b := b + 1;
      FGIntShiftRight(temp1);
   End;
   m := temp1;

   i := 0;
   ok := true;
   Randomize;
   While (i < nrtest) And ok Do
   Begin
      i := i + 1;
      Base10StringToFGInt(inttostr(Primes[Random(1227) + 1]), temp2);
      FGIntMontGomeryModExp(temp2, m, FGIntp, z);
      FGIntDestroy(temp2);
      ok1 := (FGIntCompareAbs(z, one) = Eq);
      ok2 := (FGIntCompareAbs(z, pmin1) = Eq);
      If Not (ok1 Or ok2) Then
      Begin

         While (ok And (j < b)) Do
         Begin
            If (j > 0) And ok1 Then ok := false
            Else
            Begin
               j := j + 1;
               If (j < b) And (Not ok2) Then
               Begin
                  FGIntSquaremod(z, FGIntp, temp3);
                  FGIntCopy(temp3, z);
                  ok1 := (FGIntCompareAbs(z, one) = Eq);
                  ok2 := (FGIntCompareAbs(z, pmin1) = Eq);
                  If ok2 Then j := b;
               End
               Else If (Not ok2) And (j >= b) Then ok := false;
            End;
         End;

      End
   End;

   FGIntDestroy(zero);
   FGIntDestroy(one);
   FGIntDestroy(two);
   FGIntDestroy(m);
   FGIntDestroy(z);
   FGIntDestroy(pmin1);
End;


// Розраховуємо коефіцієнти з теореми Безу, FGInt1 * a + FGInt2 * b =
GCD(FGInt1, FGInt2)

Procedure FGIntBezoutBachet(Var FGInt1, FGInt2, a, b : TFGInt);
Var
   zero, r1, r2, r3, ta, gcd, temp, temp1, temp2 : TFGInt;
Begin
   If FGIntCompareAbs(FGInt1, FGInt2) <> St Then
   Begin
      FGIntcopy(FGInt1, r1);
      FGIntcopy(FGInt2, r2);
      Base10StringToFGInt('0', zero);
      Base10StringToFGInt('1', a);
      Base10StringToFGInt('0', ta);
```

```
      Repeat
         FGIntdivmod(r1, r2, temp, r3);
         FGIntDestroy(r1);
         r1 := r2;
         r2 := r3;

         FGIntmul(ta, temp, temp1);
         FGIntsub(a, temp1, temp2);
         FGIntCopy(ta, a);
         FGIntCopy(temp2, ta);
         FGIntDestroy(temp1);

         FGIntDestroy(temp);
      Until FGIntCompareAbs(r3, zero) = Eq;

      FGIntGCD(FGInt1, FGInt2, gcd);
      FGIntmul(a, FGInt1, temp1);
      FGIntsub(gcd, temp1, temp2);
      FGIntDestroy(temp1);
      FGIntdiv(temp2, FGInt2, b);
      FGIntDestroy(temp2);

      FGIntDestroy(ta);
      FGIntDestroy(r1);
      FGIntDestroy(r2);
      FGIntDestroy(gcd);
   End
   Else FGIntBezoutBachet(FGInt2, FGInt1, b, a);
End;


// Знахродимо мультпликативне зворотнє FGInt у кінцевому кільці позитивного
порядку

Procedure FGIntModInv(Const FGInt1, base : TFGInt; Var Inverse : TFGInt);
Var
   zero, one, r1, r2, r3, tb, gcd, temp, temp1, temp2 : TFGInt;
Begin
   Base10StringToFGInt('1', one);
   FGIntGCD(FGInt1, base, gcd);
   If FGIntCompareAbs(one, gcd) = Eq Then
   Begin
      FGIntcopy(base, r1);
      FGIntcopy(FGInt1, r2);
      Base10StringToFGInt('0', zero);
      Base10StringToFGInt('0', inverse);
      Base10StringToFGInt('1', tb);

      Repeat
         FGIntDestroy(r3);
         FGIntdivmod(r1, r2, temp, r3);
         FGIntCopy(r2, r1);
         FGIntCopy(r3, r2);

         FGIntmul(tb, temp, temp1);
         FGIntsub(inverse, temp1, temp2);
         FGIntDestroy(inverse);
         FGIntDestroy(temp1);
         FGIntCopy(tb, inverse);
         FGIntCopy(temp2, tb);

         FGIntDestroy(temp);
      Until FGIntCompareAbs(r3, zero) = Eq;

      If inverse.Sign = negative Then
      Begin
         FGIntadd(base, inverse, temp);
         FGIntCopy(temp, inverse);
      End;
```

```
      FGIntDestroy(tb);
      FGIntDestroy(r1);
      FGIntDestroy(r2);
   End;
   FGIntDestroy(gcd);
   FGIntDestroy(one);
End;


// Простй комбінований тест на простоту FGIntp

Procedure FGIntPrimetest(Var FGIntp : TFGInt; nrRMtests : integer; Var ok :
boolean);
Begin
   FGIntTrialdiv9999(FGIntp, ok);
   If ok Then FGIntRabinMiller(FGIntp, nrRMtests, ok);
End;


//Розрахунок символу Лагранжа

Procedure FGIntLegendreSymbol(Var a, p : TFGInt; Var L : integer);
Var
   temp1, temp2, temp3, temp4, temp5, zero, one : TFGInt;
   i : int64;
   ok1, ok2 : boolean;
Begin
   Base10StringToFGInt('0', zero);
   Base10StringToFGInt('1', one);
   FGIntMod(a, p, temp1);
   If FGIntCompareAbs(zero, temp1) = Eq Then
   Begin
      FGIntDestroy(temp1);
      L := 0;
   End
   Else
   Begin
      FGIntDestroy(temp1);
      FGIntCopy(p, temp1);
      FGIntCopy(a, temp2);
      L := 1;
      While FGIntCompareAbs(temp2, one) <> Eq Do
      Begin
         If (temp2.Number[1] Mod 2) = 0 Then
         Begin
            FGIntSquare(temp1, temp3);
            FGIntSub(temp3, one, temp4);
            FGIntDestroy(temp3);
            FGIntDivByInt(temp4, temp3, 8, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok1 := false Else ok1 := true;
            FGIntDestroy(temp3);
            FGIntDestroy(temp4);
            If ok1 = true Then L := L * (-1);
            FGIntDivByIntBis(temp2, 2, i);
         End
         Else
         Begin
            FGIntSub(temp1, one, temp3);
            FGIntSub(temp2, one, temp4);
            FGIntMul(temp3, temp4, temp5);
            FGIntDestroy(temp3);
            FGIntDestroy(temp4);
            FGIntDivByInt(temp5, temp3, 4, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok2 := false Else ok2 := true;
            FGIntDestroy(temp5);
            FGIntDestroy(temp3);
            If ok2 = true Then L := L * (-1);
            FGIntMod(temp1, temp2, temp3);
```

```
            FGIntCopy(temp2, temp1);
            FGIntCopy(temp3, temp2);
         End;
      End;
      FGIntDestroy(temp1);
      FGIntDestroy(temp2);
   End;
   FGIntDestroy(zero);
   FGIntDestroy(one);
End;


// Розрахунок квадратичного корня за модулем простого числа
// SquareRoot^2 mod Prime = Square

Procedure FGIntSquareRootModP(Square, Prime : TFGInt; Var SquareRoot : TFGInt);
Var
   one, n, b, s, r, temp, temp1, temp2, temp3 : TFGInt;
   a, L, i, j : longint;
Begin
   Base2StringToFGInt('1', one);
   Base2StringToFGInt('10', n);
   a := 0;
   FGIntLegendreSymbol(n, Prime, L);
   While L <> -1 Do
   Begin
      FGIntAddBis(n, one);
      FGIntLegendreSymbol(n, Prime, L);
   End;
   FGIntCopy(Prime, s);
   s.Number[1] := s.Number[1] - 1;
   While (s.Number[1] Mod 2) = 0 Do
   Begin
      FGIntShiftRight(s);
      a := a + 1;
   End;
   FGIntMontgomeryModExp(n, s, Prime, b);
   FGIntAdd(s, one, temp);
   FGIntShiftRight(temp);
   FGIntMontgomeryModExp(Square, temp, Prime, r);
   FGIntDestroy(temp);
   FGIntModInv(Square, Prime, temp1);

   For i := 0 To (a - 2) Do
   Begin
      FGIntSquareMod(r, Prime, temp2);
      FGIntMulMod(temp1, temp2, Prime, temp);
      FGIntDestroy(temp2);
      For j := 1 To (a - i - 2) Do
      Begin
         FGIntSquareMod(temp, Prime, temp2);
         FGIntDestroy(temp);
         FGIntCopy(temp2, temp);
         FGIntDestroy(temp2);
      End;
      If FGIntCompareAbs(temp, one) <> Eq Then
      Begin
         FGIntMulMod(r, b, Prime, temp3);
         FGIntDestroy(r);
         FGIntCopy(temp3, r);
         FGIntDestroy(temp3);
      End;
      FGIntDestroy(temp);
      FGIntDestroy(temp2);
      If i = (a - 2) Then break;
      FGIntSquareMod(b, Prime, temp3);
      FGIntDestroy(b);
      FGIntCopy(temp3, b);
      FGIntDestroy(temp3);
```

```
    End;

    FGIntCopy(r, SquareRoot);
    FGIntDestroy(r);
    FGIntDestroy(s);
    FGIntDestroy(b);
    FGIntDestroy(temp1);
    FGIntDestroy(one);
    FGIntDestroy(n);
End;


End.
```

Файл FGIntPrimeGeneration.pas – генерація випадкових простих чисел для алгоритму RSA

```
Unit FGIntPrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, FGInt;

Procedure PrimeSearch(Var GInt : TFGInt);

Implementation


{$H+}


// Відбувається пошаговий пошук простого числа починаючи з GInt,
// якщо воно знайдено то зберігається у GInt

Procedure PrimeSearch(Var GInt : TFGInt);
Var
   temp, two : TFGInt;
   ok : Boolean;
Begin
   If (GInt.Number[1] Mod 2) = 0 Then GInt.Number[1] := GInt.Number[1] + 1;
   Base10StringToFGInt('2', two);
   ok := false;
   While Not ok Do
   Begin
      FGIntAdd(GInt, two, temp);
      FGIntCopy(temp, GInt);
      FGIntPrimeTest(GInt, 4, ok);
   End;
   FGIntDestroy(two);
End;

End.
```

Файл about.pas - довідка про програму

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls;
type
  Tfrm_about = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frm_about: Tfrm_about;
implementation
{$R *.dfm}
procedure Tfrm_about.Button1Click(Sender: TObject);
begin
frm_about.Close;
end;
procedure Tfrm_about.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Дослідження та програмна реалізація системи проектування
сервісів автентифікації');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Буравченко К.О.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Середа Максим Леонідович ');
Memo1.Lines.Add('                  гр. КІ-21М-1,4');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2022');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;
end.
```