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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу”

Виконав здобувач вищої освіти
IV курсу, групи КБ-19
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Усатов Г.С.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Усатову Глібу Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 12-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Усатов Г.С.
(прізвище та ініціали)

АНОТАЦІЯ

Усатов Г.С. Програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для автентифікації користувача хмарного сервісу.

Метою розробки є програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу.

Результат роботи – програмна реалізація системи кібербезпеки для автентифікації користувача хмарного сервісу.

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 Sydney.

Ключові слова: кібербезпека, автентифікаця хмарний сервіс

ABSTRACT

Usatov G.S. Cyber security system software for cloud service user authentication. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system for cloud service user authentication.

The purpose of the development is the software of the cyber security system for the authentication of the user of the cloud service.

The result of the work is the software implementation of the cyber security system for cloud service user authentication.

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 Sydney environment.

Keywords: cyber security, authentication, cloud service

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 теми. Платформа Системи кібербезпеки для автентифікації користувача хмарного сервісу дозволяє користувачам автентифікуватися у програмах і службах, як-от програми SaaS, мобільні/веб-програми, ігри, API тощо.

Платформа Системи кібербезпеки для автентифікації користувача хмарного сервісу забезпечує безпечну та просту автентифікацію, якщо ви створюєте службу в Хмарному сервісі, на власному сервері чи на іншій платформі.

Системи кібербезпеки для автентифікації користувача хмарного сервісу надає серверні служби та працює з простими у використанні SDK і готовими бібліотеками інтерфейсу користувача для автентифікації користувачів у вашій програмі. Він підтримує автентифікацію за допомогою паролів, номерів телефонів, популярних федеративних постачальників ідентифікаційної інформації, таких як Хмарний сервіс, Facebook, Twitter, і будь-якого постачальника, який підтримує SAML або протокол OpenID Connect.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу.

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

– Огляд існуючих систем для автентифікації користувача хмарного сервісу.

– Дослідження системи кібербезпеки для автентифікації користувача хмарного сервісу.

– Програмна реалізація системи кібербезпеки для автентифікації користувача хмарного сервісу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для автентифікації користувача хмарного сервісу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У даному бакалаврському проекті буде розглянута користувача хмарного сервісу з використанням електронних ключів, більш конкретно з використанням як ключ – USB-накопичувач.

Більшість операцій, які ви виконуєте в Хмарному сервісі, мають бути автентифіковані. Виняток становлять лише операції над об’єктами, які дозволяють анонімний доступ. Об’єкти доступні анонімно, якщо allUsers група має READ дозвіл. До групи allUsers входять усі користувачі Інтернету.

Хмарний сервіс використовує OAuth 2.0 для автентифікації та авторизації API. Автентифікація – це процес визначення особистості клієнта. Деталі автентифікації відрізняються залежно від того, як ви отримуєте доступ до Хмарного сервісу, але поділяються на два загальні типи:

– Потік, орієнтований на сервер, дозволяє програмі безпосередньо зберігати облікові дані облікового запису служби для завершення автентифікації. Використовуйте цей потік, якщо ваша програма працює з власними даними, а не з даними користувача. Проекти Хмарного сервісу мають облікові записи служби за замовчуванням, якими можна користуватися, або можливо створити нові.

– Потік, орієнтований на користувача, дозволяє програмі отримувати облікові дані від кінцевого користувача. Користувач входить, щоб завершити автентифікацію. Використовуйте цей потік, якщо вашій програмі потрібен доступ до даних користувача.

Майте на увазі, що можливо використовувати обидва типи автентифікації разом у програмі.

1.2 Область застосування

Областю застосування розробленого програмного забезпечення, є розподілення доступу до системи на хмарному сервісі.

Авторизація – це процес визначення дозволів, які має автентифікована особа для набору вказаних ресурсів. OAuth 2.0 використовує області, щоб визначити, чи авторизовано автентифіковану особу. Програми використовують облікові дані (отримані з потоку автентифікації, орієнтованого на користувача чи сервера) разом із однією чи кількома областями, щоб запитувати маркер доступу від сервера авторизації Хмарного сервісу для доступу до захищених ресурсів. Наприклад, програма A з маркером доступу з read-only областю може лише читати, а програма B з маркером доступу з read-write областю може читати та змінювати дані. Жодна програма не може читати або змінювати списки контролю доступу до об’єктів і сегментів; лише програма з full-control областю дії може це зробити.

Автентифікація клієнтської бібліотеки

Клієнтські бібліотеки можуть використовувати облікові дані програми за замовчуванням для легкої автентифікації за допомогою API Хмарний сервіс і надсилання запитів до цих API. За допомогою облікових даних програми за замовчуванням можливо перевірити свою програму локально та розгорнути її, не змінюючи базовий код.

Хмарний сервіс

Якщо ви запускаєте свою програму на App Engine, Cloud Functions, Cloud Run або Compute Engine, середовище вже надає інформацію автентифікації облікового запису служби, тому подальше налаштування не потрібне. Для Compute Engine обсяг облікового запису служби залежить від того, як ви створили екземпляр.

Інші середовища

Щоб ініціалізувати локальне середовище розробки або виробництва, створіть обліковий запис служби Хмарному сховищі, завантажте його ключ і встановіть ХМАРНИЙ_СЕРВІС_APPLICATION_CREDENTIALS змінну середовища для використання ключа. Як альтернативу встановленню змінної середовища ви також можете використовувати ключ облікового запису служби безпосередньо в коді.

Автентифікація API

Щоб надсилати запити за допомогою OAuth 2.0 до Хмарного сервісу XML API або JSON API, додайте маркер доступу вашої програми в заголовок Authorizationкожного запиту, який вимагає автентифікації. Можливо створити маркер доступу з OAuth 2.0 Playground:

1. У OAuth 2.0 Playground клацніть **Хмарний сервіс API v1**, а потім виберіть рівень доступу для своєї програми (full_control, read_onlyабо read_write).

2. Натисніть **Авторизувати API**.

3. Коли буде запропоновано, увійдіть у свій обліковий запис Хмарний сервіс. У діалоговому вікні, що з’явиться, натисніть **«Дозволити»**.

4. На кроці 2 ігрового майданчика натисніть **«Обмінити код авторизації на токени»,** щоб отримати код авторизації, який з’явиться.

5. Скопіюйте свій маркер доступу та додайте його в Authorizationзаголовок свого запиту:

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо огляд існуючого програмного забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу.

DeviceLock

DeviceLock – сучасний програмний засіб, призначений для захисту й адміністрування локальних і мережних комп'ютерів шляхом запобігання неконтрольованих дій користувача при обміні інформацією через комп'ютерні порти й пристрої зі змінними носіями.

Використання неавторизованих USB-пристроїв являє загрозу корпоративним мережам і даним. Причому не тільки конфіденційна інформація може "піти" з корпоративної мережі через USB-порт, але й віруси або троянські програми можуть бути занесені усередину корпоративної мережі, минаючи серверні файрволи й антивіруси. Точно так само справа є із записуючими CD/DVD-приводами й з USB-пристроями.

Забезпечуючи контроль над користувачами, що мають доступ до портів і пристроїв локального комп'ютера, DeviceLock закриває потенційну уразливість у захисті простим і економічним способом. DeviceLock повністю інтегрується в підсистему безпеки Windows, функціонуючи на рівні ядра системи, і забезпечує прозорий для користувача захист.

Програмний комплекс "DeviceLock" складається із трьох компонентів: агента (DeviceLock Service), сервера (DeviceLock Enterprise Server і DeviceLock Content Security Server) і консолі керування (DeviceLock Management Console, DeviceLock Group Policy Manager або DeviceLock Enterprise Manager).

Агент DeviceLock Service, будучи основним компонентом програмного комплексу "DeviceLock", установлюється на кожний клієнтський комп'ютер, автоматично запускається при завантаженні ОС.

DeviceLock Enterprise Server – компонент, використовуваний для централізованого збору й зберігання даних тіньового копіювання й журналів аудита. DeviceLock Enterprise Server використовує MS SQL Server для зберігання даних. Для забезпечення рівномірного навантаження мережі й Хмарний сервіс допускається установка декількох екземплярів DeviceLock Enterprise Server.

DeviceLock Content Security Server – окремо ліцензуємий компонент, що включає в себе пошуковий сервер для швидкого пошуку тексту у файлах тіньового копіювання й журналах, що зберігаються на DeviceLock Enterprise Server.

Консоль керування – це інтерфейс контролю, що системний адміністратор використовує для віддаленого керування будь-якою системою, на якій установлений агент (DeviceLock Service).

DeviceLock поставляється із трьома різними консолями керування: DeviceLock Management Console (оснащення для MMC), DeviceLock Enterprise Manager і DeviceLock Group Policy Manager (інтегрований у редактор групових політик Windows). DeviceLock Management Console також використовується для керування DeviceLock Enterprise Server і DeviceLock Content Security Server.

DeviceLock дозволяє:

– контролювати доступ користувачів або груп до внутрішніх пристроїв – контролерів Wi-Fi, Bluetooth, CD– і DVD-дисководів, жорстких дисків; зовнішніх USB-носіїв, портів вводу-виводу USB, IEEE 1394, COM, LPT, IrDA; локальних і мережних принтерів;

– контролювати доступ користувачів і груп до пристроїв і портів вводу-виводу залежно від часу доби й дня тижня;

– встановлювати тип доступу "тільки читання" для змінних носіїв, дисководів, жорстких дисків і CD/ DVD-приводів;

– захищати диски й змінні носії від випадкового або навмисного форматування;

– задавати особистий для кожного користувача або групи список пристроїв, доступ до яких буде завжди дозволений. Пристрої можуть ідентифікуватися по моделі й по унікальному серійному номері;

– ідентифікувати на основі аналізу записаних даних CD/ DVD-диск і дозволяти його використання, навіть якщо сам CD/ DVD-привод заблокований;

– надавати користувачеві тимчасовий доступ до пристроїв по спеціальному буквено-цифровому коді, переданому по телефоні адміністратором, при відсутності мережного підключення агента;

– протоколювати всі дії користувачів із пристроями й файлами (копіювання, читання, видалення й т.п.), зі змінами настроювань агента, його часу запуску й зупинки;

– зберігати для кожного користувача або групи точну копію даних, переданих на зовнішні пристрої або через різні порти (тіньове копіювання). Точні копії всіх файлів і даних зберігаються в SQL-базі даних (БД);

– установлювати "контентно-залежні" правила на рівні файлових операцій. Це дозволяє дозволяти й забороняти доступ до певних типів файлів поза залежністю від установлених на пристрій дозволів. Контентно-залежні правила можна використовувати для включення й відключення тіньового копіювання для заданих типів файлів. Визначення типів файлів засновано на сигнатурному методі й не залежить від розширення файлу. Підтримується більше 3000 різних типів файлів;

– централізовано зберігати журнали аудита й тіньового копіювання. Для централізованого збору й зберігання даних тіньового копіювання й журналів аудита використовується SQL-сервер і компонент DeviceLock Enterprise Server. Можна встановити кілька екземплярів DeviceLock Enterprise Server у мережі для рівномірного розподілу навантаження;

– забезпечувати необхідний рівень захисту, навіть якщо користувачі в мережі мають адміністративні привілеї на локальних комп'ютерах. Коли захист DeviceLock включений, ніхто, крім авторизованих адміністраторів, не може управляти агентом або обійти його захист;

– забезпечувати за допомогою системи віддаленого керування доступ до всіх можливих функцій агента з робочого місця адміністратора DeviceLock. Інструментом адміністратора DeviceLock є оснащення DeviceLock Management Console для Microsoft Management Console зі стандартним інтерфейсом Windows. Крім того, для керування DeviceLock у мережах, де не використовується Active Directory, передбачена консоль DeviceLock Enterprise Manager із власним інтерфейсом;

– управляти агентом через групові політики Windows у домені Active Directory за допомогою стандартного оснащення Group Policy, що входить до складу серверних операційних систем починаючи з Windows 2000 Server. Повна інтеграція в групові політики Windows дозволяє автоматично встановлювати DeviceLock на нові комп'ютери, що підключаються до корпоративної мережі, і здійснювати настроювання для нових комп'ютерів в автоматичному режимі;

– забезпечувати підтримувати контекстно-контекстно-залежну систему політик безпеки (залежно від наявності або відсутності підключення комп'ютера до корпоративної мережі, доступності контролера домену, доступності DeviceLock Enterprise Server);

– вибирати комп'ютери прямо зі служб каталогів LDAP;

– формувати звіти по встановлених настроюваннях і по пристроях (USB, IEEE 1394), які використовують користувачі на своїх клієнтських комп'ютерах;

– формувати звіти (у т.ч. і графічні) по встановлених настроюваннях і по пристроях (USB, IEEE 1394), які використовують користувачі на своїх клієнтських комп'ютерах;

– здійснювати контроль, аудит і тіньове копіювання для локальних і мережних принтерів. Контролювати відправлення документів на друк по

користувачах і часу. Підтримуються всі інтерфейси (USB, LPT, мережа). Включити аудита й тіньове копіювання для документів, що друкуються.

eToken Network Logon

eToken Network Logon – програмне забезпечення посилення функцій безпеки операційних систем Microsoft Windows.

Програмне забезпечення посилення функцій безпеки операційних систем Microsoft Windows – eToken Network Logon 5 – призначено для входу на робочу станцію й у домен Microsoft Windows з використанням USB-ключів і смарт-карт eToken.

eToken Network Logon забезпечує двофакторну автентифікацію користувачів і адміністраторів робочих станцій з використанням eToken, автоматично блокує робочу станцію при від'єднанні eToken. Застосування eToken Network Logon дозволяє вирішити проблему "слабких" паролів і автоматизувати виконання користувачами вимог регламентів ІБ.

Можливості:

– eToken Network Logon може бути встановлений на робочі станції під керуванням Microsoft Windows Server, об'єднані в робочу групу або домен Windows. Після установки сертифікованої версії eToken Network Logon стандартне запрошення для входу в Microsoft Windows заміняється новим, котре змінює порядок входу користувача в систему.

– Для забезпечення можливості входу користувача на робочу станцію або в домен Windows адміністратор створює в пам'яті електронного ключа eToken профіль користувача. Профіль користувача містить необхідні реєстраційні дані користувача (ім'я користувача, його пароль, найменування робочої станції або домену). При завданні пароля користувача eToken Network Logon автоматично генерує складний пароль, що відповідає вимогам діючої парольної політики, виконує його установку в операційній системі робочої станції або в домене й зберігає згенерований пароль у захищеній пам'яті eToken.

– Двофакторна автентифікація користувачів і адміністраторів на робочій станції й у мережі Windows за допомогою USB-ключів або смарт-карт eToken. Сертифікована версія eToken Network Logon дозволяє не просто зберегти реєстраційні дані користувача в пам'яті електронного ключа eToken, але й захистити їхнім паролем користувача eToken. У випадку втрати або крадіжки електронного ключа eToken це не приведе до компрометації реєстраційних даних користувача або адміністратора робочої станції.

– Автоматична генерація складних паролів, що відповідають вимогам діючої парольної політики, їхня установка в операційній системі й збереження в захищеній пам'яті eToken для наступного використання.

– Автоматичне блокування консолі робочої станції при від'єднанні eToken.

– Просте й зручне для кінцевого користувача використання паролів, що відповідають вимогам парольної політики й зберігаються в пам'яті eToken (користувачеві досить приєднати eToken до робочої станції й увести пароль користувача eToken – при цьому користувач позбувається від необхідності запам'ятовувати складний пароль).

– Рішення проблеми "слабких" паролів. Після установки продукту можна використовувати збережені в пам'яті електронного ключа eToken складні паролі (автоматично згенеровані й невідомі користувачеві). Пароль перестає бути "слабким" – виключається ризик його підбора зловмисником. При можливій зміні пароля користувачем виключається ризик завдання їм "слабкого" пароля, так як пароль не вводиться із клавіатури, а генерується автоматично – отже, виключаються ризики підглядання пароля або його перехоплення шпигунським ПЗ. Крім того, користувач не повинен пам'ятати пароль – виключаються випадки його забування й записування на папері.

– Підвищення загального рівня інформаційної безпеки в організації. При використанні сертифікованої версії eToken Network Logon на робочій станції може бути заборонене використання стандартного методу автентифікації в Windows (по ім'ю користувача й паролю).

Переваги:

– Відмова від вводу паролів вручну. При вході в систему користувач ніколи не вводить пароль. Це виключає ризики підглядання пароля або його перехоплення при уведенні із клавіатури.

– Практична можливість і зручність застосування складних паролів. Оскільки користувач не повинен уводити пароль вручну, сам пароль може бути більш довгим й складнішим, ніж користувач може запам'ятати. Максимальна довжина паролів, автоматично генеруємих у сертифікованій версії eToken Network Logon, становить 64 символ.

– Використання паролів, невідомих користувачеві. Сертифікована версія eToken Network Logon дозволяє генерувати паролі заданої довжини (64 символу), зберігати їх у пам'яті електронного ключа eToken і підставляти в сховище облікових даних таким чином, що користувач навіть не знає свого пароля, а тому не може записати його на папері (повідомити кого-небудь і т.п.) і тим самим скомпрометувати його.

– Простота й зручність для користувачів. Спосіб автентифікації, застосовуваний у сертифікованій версії eToken Network Logon, зручніше для користувачів, чим стандартні способи.

Вимоги до якості паролів користувача eToken не настільки високі, як вимоги до складності паролів. Тому при двофакторної автентифікації вводити пароль користувача eToken простіше, ніж без такої вводити складний і довгий пароль. Якщо електронний ключ eToken підключений до комп'ютера, необов'язково відключати його й підключати знову. Сертифікована версія eToken Network Logon дозволяє в такому випадку нажати на клавіатурі комбінацію клавіш CTRL+L, увести пароль користувача eToken і ввійти в систему, не підключаючи повторно електронний ключ eToken до робочої станції.

Ексклюзивні особливості:

– Сертифікована версія eToken Network Logon – універсальний продукт, якому можна використовувати на ізольованій робочій станції, у невеликій робочій групі, у простій або складній доменній інфраструктурі.

– Використовувані в складі продукту сертифіковані електронні ключі eToken можуть також бути використані для зберігання ключової інформації сертифікованих СКЗІ (Криптопро CSP, Signal-COM CSP, Домен-К і др.), інший ключовий і автентифікаційної інформації користувача (паролі, коди доступу й т.д.).

– Сертифіковану версію eToken Network Logon можна інтегрувати з eToken TMS (Token Management System) – системою керування життєвим циклом електронних ключів eToken. Це дозволить управляти електронними ключами eToken і профілями користувачів централізовано.

MultiKey

MultiKey – універсальний збірний емулятор електронних ключів. Призначений для більш-менш точного відтворення (емуляції) дій електронних ключів при роботі із додатком, щозахищається.

Ціль використання – розробка (доробка, тестування) захисних механізмів програмного забезпечення, що виключають використання емуляторів для запуску ПЗ без апаратних ключів захисту або яке робить цей процес надзвичайно трудомістким.

На даний момент MultiKey підтримує емуляцію таких типів електронних ключів:

– Hasp3/4, Hasp HL, Hasp SRM.

– Hardlock.

– Sentinel superpro, ultrapro.

– Guardant Stealth I, Stealth II.

– Dinkey.

Електронний ключ HASP HL Max

Переваги електронних ключів HASP HL Max

– Захист до 233 додатків або модулів за допомогою одного ключа.

– Унікальний ID номер.

– Високий рівень захисту (алгоритм AES/128).

– Захищена пам'ять (4 KB на читання/запис; 2 KB тільки на читання).

– Можливість дистанційного перепрограмування ключа.

– Кроссплатформеність (Windows, Linux, Mac).

HASP HL Max – ідеальне рішення для захисту програм, що складаються з багатьох модулів і компонентів. Обсяг захищеної пам'яті дозволяє розроблювачеві встановити ліцензійні обмеження на 233 різні додатки, модулі або функції.

Використання одного ключа HASP HL Max для захисту всього набору додатків дає можливість розроблювачеві реалізувати гнучку схему продажів. За допомогою Business Studio Application, що входить до складу Комплекту розроблювача, можна дистанційно (наприклад, через Інтернет) змінювати моделі ліцензування додатків, захищених HASP HL Max. При цьому ніяких змін у захист додатка вносити не потрібно. Менеджери по продукту, відповідальні за його ліцензування, можуть у будь-який момент розширити функціональність захищеного додатка, включити/виключити ті або інші модулі програмного забезпечення, обновити версію й т.д. Всі відновлення пам'яті підписуються за допомогою цифрового підпису (RSA/1024), що виключає можливість внесення несанкціонованих змін.

Кожний ключ HASP HL Max містить унікальний ID номер, що дозволяє зробити захист кожної копії додатка унікальним. Знання ID номера ключа дає можливість повністю контролювати продажі й відновлення захищеного ПЗ.

Програмний комплекс "Shield Channel Client" (ПК "SCC")

Програмний комплекс "Shield Channel Client" призначений для організації безпечного доступу віддалених (мобільних) користувачів у захищені віртуальні корпоративні мережі (ЗВКМ), створювані на базі СКЗІ "Криптоканал" (ПАК "SC-FW").

ЗВКМ являє собою об'єднання територіально-віддалених локальних обчислювальних мереж (сегментів ЗВКМ) у віртуальну мережу, що використовує як транспортне середовище публічні мережі (включаючи мережу Інтернет, відомчі мережі й т.п.) і цілісність, що забезпечує, і безпеку переданих даних. У точці підключення сегмента ЗВКМ до публічної мережі встановлюється ПАК "SC-FW", що складається із двох серверів – внутрішнього й зовнішнього. Зовнішній шлюз має вихід у зовнішню (публічну) мережу, внутрішній – у внутрішню (закриту) мережу

ПК "Shield Channel Client" встановлюється на окремому комп'ютері під керуванням операційної системи Windows, підключеному до публічної мережі, і реалізує захищену взаємодію із сегментами ЗВКМ. Таким чином, користувач одержує можливість підключатися до корпоративних ресурсів з віддаленого комп'ютера.

Для створення захищеного каналу ПК "Shield Channel Client" використовує архітектуру й протоколи IPSec. Для шифрування переданих даних і додаткової автентифікації й перевірки цілісності використовується протокол ESP – Encapsulating Security Payload (вкладені захищені передані дані). Для шифрування даних і перевірки цілісності ПК "Shield Channel Client" використовує ключі, що розподіляються динамічно.

ПК "Shield Channel Client" має власні криптографічні модулі, що реалізують алгоритми шифрування, ДСТУ 28147:2009 і алгоритми автентифікації й перевірки цілісності ДСТУ 28147:2009 (імітовставка).

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки 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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи кібербезпеки для автентифікації користувача хмарного сервісу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки 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 кібербезпеки 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

Автентифікація за допомогою ключів API

Більшість API Хмарних сховищ не підтримують ключі API. Перш ніж використовувати цей метод автентифікації, переконайтеся, що API, який ви хочете використовувати, підтримує ключі API.

Знайомство з ключами API

Коли ви використовуєте ключ API для автентифікації в API, ключ API не ідентифікує принципала та не надає жодної інформації про авторизацію. Ключ API пов’язує запит із проектом Хмарного сховища для цілей виставлення рахунків і квот. Оскільки ключі API не ідентифікують абонента, вони часто використовуються для доступу до загальнодоступних даних або ресурсів.

Багато API Хмарних сховищ не приймають ключі API для автентифікації. Перегляньте документацію автентифікації для служби або API, які ви хочете використовувати, щоб визначити, чи підтримують вони ключі API.

Ключ API містить такі компоненти, які ви використовуєте для керування та використання ключа:

Рядок

Рядок ключа API є зашифрованим рядком, наприклад, AIzaSyDaGmWKa4JsXZ-HjGw7ISLn_3namBGewQe. Коли ви використовуєте ключ API для автентифікації, ви завжди використовуєте рядок ключа. Ключі API не мають пов’язаного файлу JSON.

ID

Ідентифікатор ключа API використовується інструментами адміністрування Хмарному сховищі для унікальної ідентифікації ключа. Ідентифікатор ключа не можна використовувати для автентифікації.

Ідентифікатор ключа можна знайти в URL-адресі сторінки редагування ключа в консолі Хмарного сховища. Ви також можете отримати ідентифікатор ключа, використовуючи Хмарному сховищі CLI для переліку ключів у вашому проекті.

Відображуване ім'я

Відображуване ім’я – це необов’язкове описове ім’я для ключа, яке можна встановити під час створення або оновлення ключа.

Щоб керувати ключами API, ви повинні мати роль адміністратора ключів API (roles/serviceusage.apiKeysAdmin) у проекті.

Створіть ключ API

Щоб створити ключ API, скористайтеся одним із таких варіантів:

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть **«Створити облікові дані»**, а потім виберіть **«Ключ API»** у меню.

У діалоговому вікні створення ключа API відображається рядок для вашого щойно створеного ключа.

Скопіюйте рядок ключа та зберігайте його в безпеці. Якщо ви не використовуєте ключ тестування, який плануєте видалити пізніше, додайте обмеження для ключа програми та API.

Використовуйте ключ API

Якщо API підтримує використання ключів API, можливо використовувати ключі API для автентифікації цього API. Ви використовуєте ключі API із запитами REST і клієнтськими бібліотеками, які їх підтримують.

Використання ключа API з REST

Можливо передати ключ API у виклик REST API як параметр запиту в такому форматі. Замініть *API_KEY*рядок ключа вашого ключа API.

Крім того, можливо використовувати x-goog-api-keyзаголовок, щоб передати свій ключ. Цей заголовок потрібно використовувати із запитами gRPC.

```
curl -X POST \
    -H "Authorization: Bearer $(gcloud auth print-access-token)" \
```

```
-H "X-goog-api-key: API_KEY" \
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
"https://translation.com.ua/language/translate/v2"
```

Використання ключа API з клієнтськими бібліотеками

Підтримка клієнтської бібліотеки для ключів API залежить від мови.

Python

У цьому прикладі використовується Cloud Natural Language API, який підтримує ключі API для автентифікації, щоб продемонструвати, як можна надати ключ API для бібліотеки.

Щоб запустити цей зразок, ви повинні інсталювати Python Client for Natural Language API та Python Client for API Keys API.

```
auth/api-client/authenticate_with_api_key.py
from cloud import language_v1
def authenticate_with_api_key(quota_project_id: str, api_key_string: str) -
> None:
    """
    Authenticates with an API key for Language service.
    TODO(Developer): Replace these variables before running the sample.
    Args:
        quota_project_id: project id that should be used for quota and
billing purposes.
        api_key_string: The API key to authenticate to the service.
    """
    # Initialize the Language Service client and set the API key and the
quota project id.
    client = language_v1.LanguageServiceClient(client_options={"api_key":
api_key_string,

"quota_project_id": quota_project_id})
    text = "Hello, world!"
    document = language_v1.Document(
        content=text, type_=language_v1.Document.Type.PLAIN_TEXT
    )
    # Make a request to analyze the sentiment of the text.
    sentiment = client.analyze_sentiment(
        request={"document": document}
    ).document_sentiment
    print(f"Text: {text}")
```

```
print(f"Sentiment: {sentiment.score}, {sentiment.magnitude}")
print("Successfully authenticated using the API key")
```

Захистіть ключ API

Коли ви використовуєте ключі API у своїх програмах, переконайтеся, що вони зберігаються в безпеці під час зберігання та передачі. Публічне оприлюднення ваших ключів API може призвести до неочікуваних стягнень з вашого облікового запису. Щоб захистити ваші ключі API, дотримуйтеся наведених нижче практичних порад.

– Додайте обмеження для ключа API до свого ключа. Додавши обмеження, можливо обмежити способи використання ключа API, зменшуючи вплив скомпрометованого ключа API.

– Видаліть непотрібні ключі API, щоб мінімізувати ризик атак.

– Періодично створюйте ключі API заново.

Періодично створюйте нові ключі API, видаляйте старі ключі та оновлюйте свої програми для використання нових ключів API.

Застосувати обмеження ключів API

За замовчуванням ключі API не обмежені. Необмежені ключі небезпечні, оскільки ними може користуватися будь-хто з будь-якого місця. Для виробничих додатків слід установити як обмеження додатків, так і обмеження API.

Додати обмеження до програми

Обмеження програм визначають, які веб-сайти, IP-адреси чи програми можуть використовувати ключ API.

Одночасно можна застосувати лише один тип обмеження програми.

HTTP реферери

Щоб обмежити веб-сайти, які можуть використовувати ваш ключ API, ви додаєте одне або кілька обмежень HTTP-реферера.

Субдомен або шлях можна замінити символом підстановки ( ), але не можна вставляти символ підстановки в середину URL-адреси. Наприклад, *.Хмарний сервіс.com є дійсним і приймає всі сайти, які закінчуються на.Хмарний сервіс.com. Однак mysubdomain*.Хмарний сервіс.comце не є дійсним

обмеженням.

Номери портів можна включати в обмеження HTTP реферерів. Якщо вказати номер порту, відповідатимуть лише запити, що використовують цей порт. Якщо не вказати номер порту, запити з будь-якого номера порту збігаються.

До ключа API можна додати до 1200 посилань HTTP.

Щоб обмежити ключ API певними веб-сайтами, скористайтеся одним із наведених нижче варіантів.

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть назву ключа API, який потрібно обмежити.

3. У розділі **«Обмеження програми»** виберіть **HTTP-реферери**.

4. Для кожного обмеження, яке потрібно додати, натисніть **«Додати елемент»**, введіть обмеження та натисніть **«Готово»**.

5. Натисніть **«Зберегти»**, щоб зберегти зміни та повернутися до списку ключів API.

IP-адреси

Можливо вказати одну або кілька IP-адрес абонентів, наприклад веб-сервера або завдання cron, яким дозволено використовувати ваш ключ API. Можливо вказати IP-адреси в будь-якому з наступних форматів:

– IPv4 (198.51.100.1).

– IPv6 (2001:db8::1).

– Підмережа з використанням нотації CIDR (198.51.100.0/24, 2001:db8::/64).

Використання localhostне підтримується для обмежень сервера.

Щоб обмежити ключ API певними IP-адресами, скористайтеся одним із наведених нижче варіантів.

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть назву ключа API, який потрібно обмежити.

3. У розділі **«Обмеження програми»** виберіть **IP-адреси**.

4. Для кожної IP-адреси, яку потрібно додати, натисніть **«Додати елемент»**, введіть адресу та натисніть **«Готово»**.

5. Натисніть **«Зберегти»**, щоб зберегти зміни та повернутися до списку ключів API.

Програми для Android

Можливо обмежити використання ключа API певними програмами Android. Ви повинні вказати назву пакета та 20-байтовий відбиток сертифіката SHA-1 для кожної програми.

Щоб обмежити свій ключ API однією чи кількома програмами Android, скористайтеся одним із наведених нижче варіантів.

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть назву ключа API, який потрібно обмежити.

3. У розділі **«Обмеження програм»** виберіть **«Програми Android»**.

4. Для кожної програми Android, яку потрібно додати, клацніть **Додати елемент** і введіть назву пакета та відбиток сертифіката SHA-1, а потім клацніть **Готово**.

5. Натисніть **«Зберегти»**, щоб зберегти зміни та повернутися до списку ключів API.

Програми для iOS

Можливо обмежити використання ключа API певними програмами для iOS, надавши ідентифікатор пакета для кожної програми.

Щоб обмежити свій ключ API однією чи кількома програмами iOS, скористайтеся одним із наведених нижче варіантів.

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть назву ключа API, який потрібно обмежити.

3. У розділі **«Обмеження програм»** виберіть **«Програми iOS»**.

4. Для кожної програми iOS, яку ви хочете додати, клацніть **Додати елемент** і введіть ідентифікатор пакета, а потім клацніть **Готово**.

5. Натисніть **«Зберегти»**, щоб зберегти зміни та повернутися до списку ключів API.

Додайте обмеження API

Обмеження API визначають, які API можна викликати за допомогою ключа API.

Перш ніж ви зможете вказати API для обмеження API, API має бути ввімкнено для вашого проекту. Щоб увімкнути API, перейдіть на інформаційну панель API.

Щоб додати обмеження API, скористайтеся одним із таких варіантів:

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть назву ключа API, який потрібно обмежити.

3. У розділі **обмежень API** натисніть **Restrict key**.

4. Виберіть усі API, для доступу до яких використовуватиметься ваш ключ API.

5. Натисніть **«Зберегти»**, щоб зберегти зміни та повернутися до списку ключів API.

Отримати інформацію про проект із ключового рядка

Можливо визначити, з яким проектом Хмарному сховищі пов’язаний ключ API, за його рядком.

Замініть *KEY_STRING* рядок ключа, для якого потрібна інформація про проект.

1. В одному з наведених нижче середовищ розробки налаштуйте gcloud CLI:

– **Cloud Shell**: щоб використовувати онлайн-термінал із уже налаштованим інтерфейсом командного рядка gcloud, активуйте Cloud Shell.

Активуйте Cloud Shell на цій сторінці

Унизу цієї сторінки починається сеанс Cloud Shell і відображається підказка командного рядка. Ініціалізація сеансу може зайняти кілька секунд.

– **Локальна оболонка**: щоб використовувати локальне середовище розробки, встановіть та ініціалізуйте gcloud CLI.

2. Ви використовуєте команду пошуку gcloud beta services api-keys, щоб отримати ідентифікатор проекту з рядка ключа.

gcloud beta services api-keys lookup *KEY_STRING*

Відновлення ключа API

Якщо ви помилково видалили ключ API, можливо скасувати видалення (відновити) цей ключ протягом 30 днів після видалення ключа. Через 30 днів ви не зможете відновити ключ API.

КонсольgcloudВІДПОЧИНОК

1. У консолі Хмарного сховища перейдіть на сторінку **облікових даних**:

Перейдіть до облікових даних

2. Натисніть **Відновити видалені облікові дані**.

3. Знайдіть видалений ключ API, який потрібно відновити, і натисніть **«Відновити»**.

Поширення відновлення видалення ключа API може зайняти кілька хвилин. Після розповсюдження відновлений ключ API відображається в списку ключів API.

Опитування довготривалих операцій

Ключі API Методи API використовують довгострокові операції. Якщо ви використовуєте REST API для створення та керування ключами API, об’єкт операції повертається з початкового запиту методу. Ім’я операції використовується для опитування тривалої операції. Коли тривалий запит завершується, опитування повертає дані з тривалого запиту.

Щоб опитувати тривалу операцію API Keys API, ви використовуєте operations.getметод.

Замінити *OPERATION_NAME* назвою операції, яку повертає

довгострокова операція. Наприклад, operations/akmf.p7-358517206116-cd10a88a-7740-4403-a8fd-979f3bd7fe1c.

```
curl -X GET \
    -H "Authorization: Bearer $(gcloud auth print-access-token)" \
    -H "Content-Type: application/json; charset=utf-8" \
    "https://apikeys.apis.com/v2/OPERATION_NAME"
```

Обмеження на ключі API

Можливо створити до 300 ключів API для кожного проекту. Цей ліміт є системним і не може бути змінений за допомогою запиту на збільшення квоти.

Якщо потрібно більше ключів API, ви повинні використовувати більше одного проекту.

3.2 Розробка структурної схеми

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 хмарному сервісі 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 Хмарний сервіс, 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 Хмарний сервіс 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 Хмарний сервіс 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.

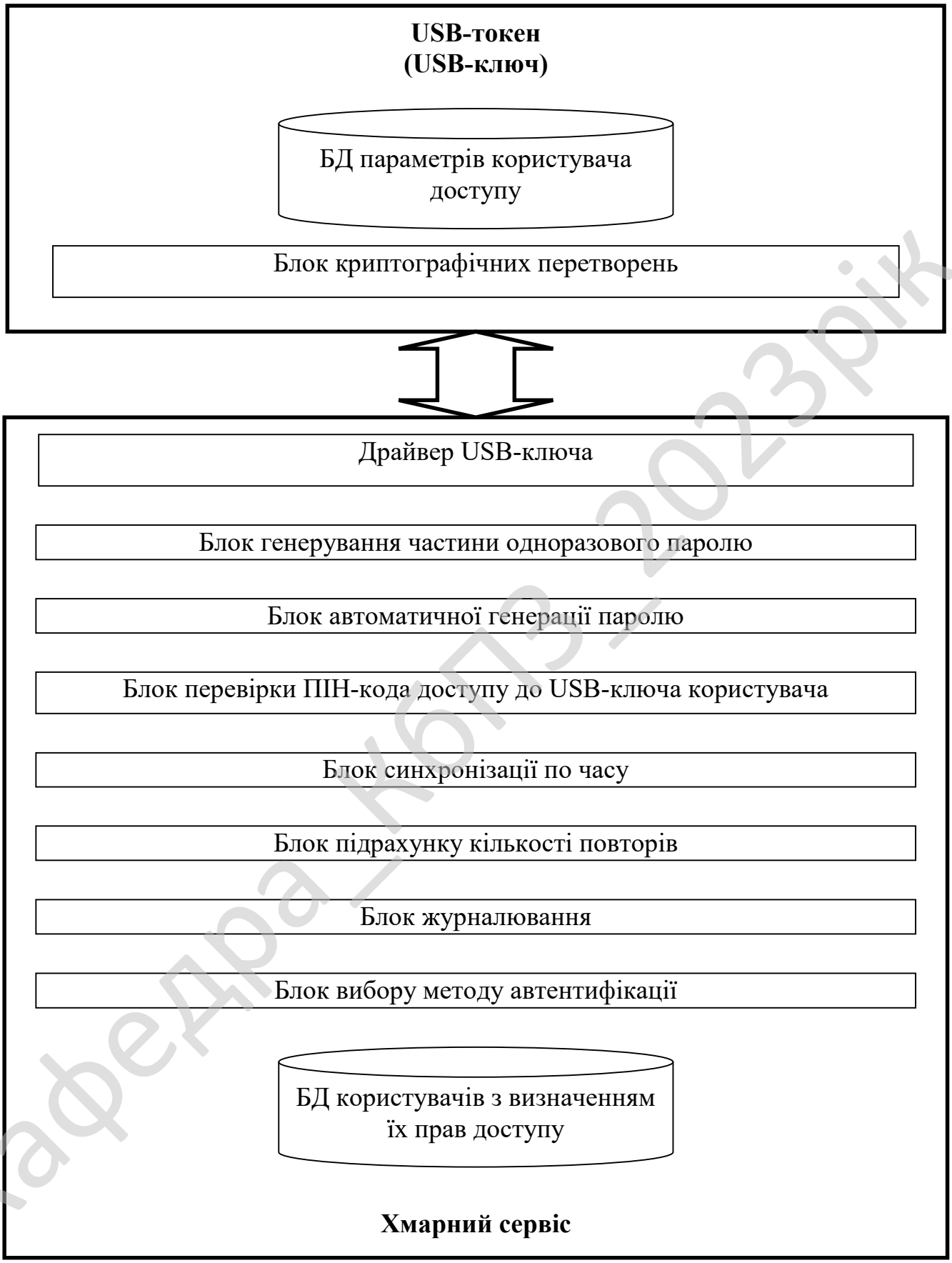

Рисунок 3.1 – Структурна схема системи

Кількість повторів обмежується трьома спробами.

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

Облікові дані облікового запису користувача

Використовуйте облікові дані облікового запису користувача для автентифікації, коли вашій програмі потрібен доступ до даних від імені користувача; інакше використовуйте облікові дані облікового запису служби. Ось приклади сценаріїв, у яких можна використовувати облікові дані користувача:

– Веб-серверні програми.

– Встановлені та настільні програми.

– Мобільні додатки.

– JavaScript на стороні клієнта.

– Програми на пристроях з обмеженим входом.

Автентифіковані завантаження браузера

Автентифіковані завантаження браузера використовують автентифікацію на основі файлів cookie. Автентифікація на основі файлів cookie просить користувачів увійти у свій обліковий запис Хмарний сервіс, щоб ідентифікувати свою особу. Зазначений обліковий запис Хмарний сервіс повинен мати відповідний дозвіл для завантаження об’єкта.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2.

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

1. 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.

2. Ініціалізаційний – право форматувати USB-ключ.

3. 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.

4. 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.

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.

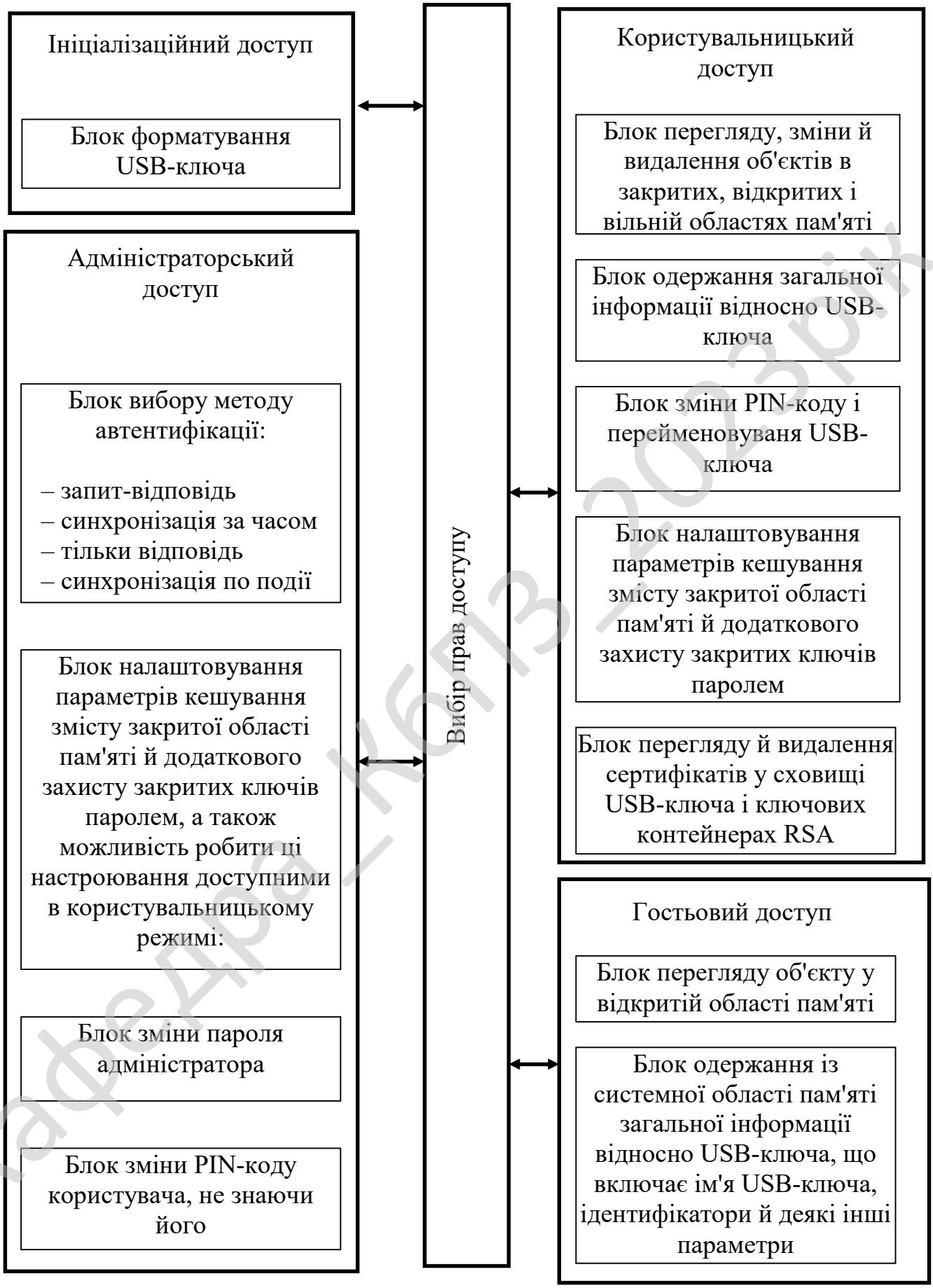

Рисунок 3.2 – Функціональна схема системи

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.

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.com.ua.

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 2000/XP/2003/Vista/2008/7 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 2000/XP/2003/Vista/2008/7, 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 2000/XP/2003/Vista/2008/7 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 тої 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 2000/XP/2003/Vista/2008/7 за допомогою USB-ключа із записаним у пам'яті паролем.

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.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

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.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. З нього видно, що процеси, які відбуваються у системі, взаємодіють наступним чином.

Спершу запускається процес початку роботи системи. Він взаємодіє з процесом виведення вікна входу у систему.

Процес виведення вікна входу у систему взаємодіє з процесом автентифікації.

На цьому етапі користувач проходить автентифікацію, щоб мати можливість працювати з комп'ютером з визначеними правами доступу.

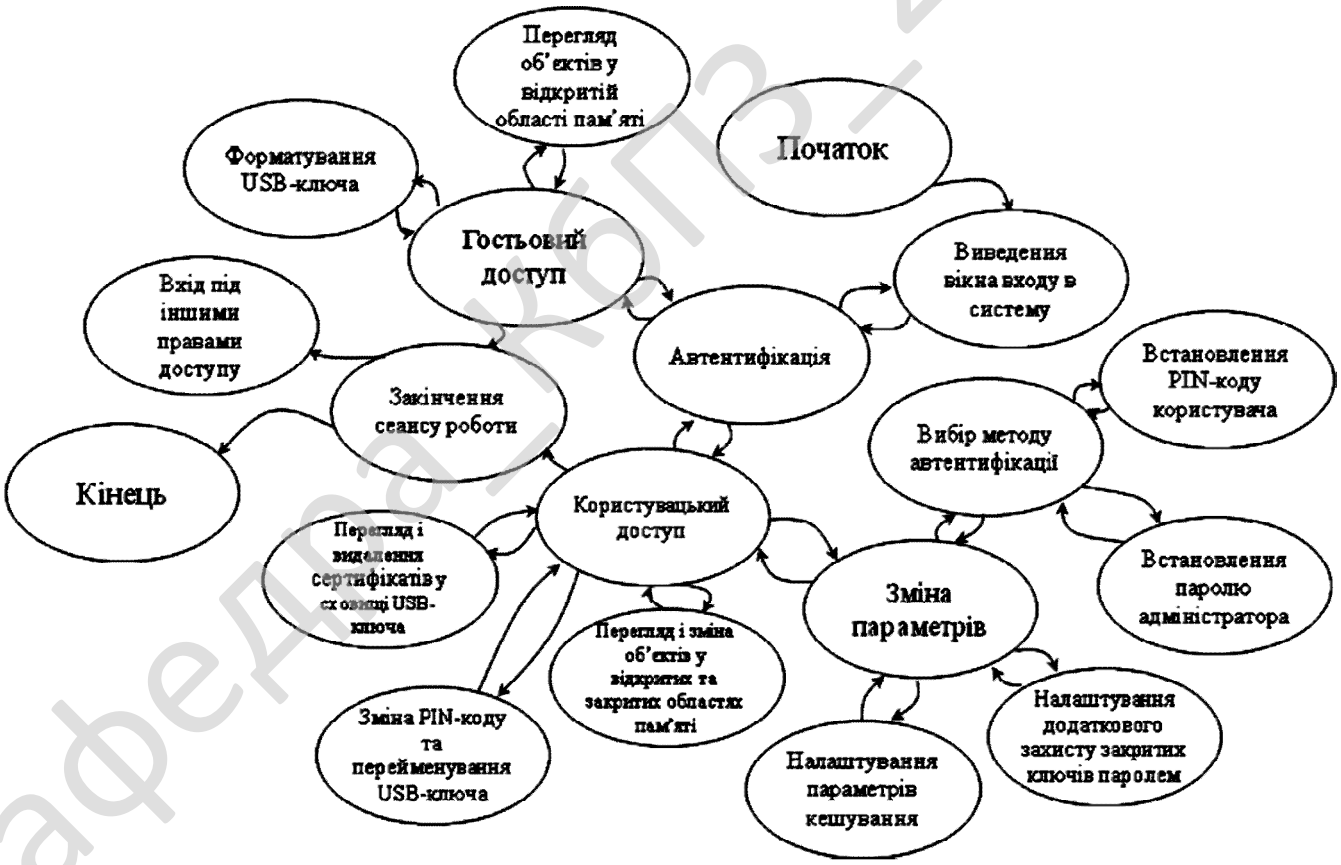

Рисунок 3.3 – Діаграма процесів системи

Процес автентифікації взаємодіє з наступними процесами:

– Процес користувальницького доступу.

– Процес гостьового доступу.

– Процес закінчення сеансу роботи.

Процес користувальницького доступу взаємодіє з наступними процесами:

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

– Процес зміни ПІН-коду та перейменування USB-ключа.

– Процес перегляду й зміни об’єктів у відкритих та закритих областях пам’яті.

– Процес зміни параметрів.

Останній процес, у свою чергу взаємодіє з наступними процесами:

– Процес налаштування параметрів кешування.

– Процес налаштування додаткового захисту закритих ключем паролем.

– Процес вибору методу автентифікації.

Процес вибору методу автентифікації взаємодіє з наступними процесами:

– Процес встановлення ПІН-коду користувача.

– Процес встановлення ПІН-коду адміністратора

Процес гостьового доступу взаємодіє з наступними процесами:

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

– Процес форматування USB-ключа.

Процес закінчення сеансу роботи взаємодіє з наступними процесами:

– Процес входу під іншими правами доступу.

– Процес закінчення роботи програми.

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

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається вибір методу автентифікаці.

Наступним кроком є налаштування USB-ключа.

За цим кроком відбувається налаштування параметрів кешування.

Крім цих налаштувань відбувається налаштування додаткового захисту закритих ключів паролем.

Після цього визначається під якими правами зареєструвався користувач: під правами користувача, або під правами гостя.

Якщо він зайшов у систему під правами користувача, тоді перевіряється правильність введення ПІН-коду доступу до USB-ключа користувача.

Якщо ПІН-код дійсний, тоді відбувається виконання наступних дій:

– Переглядаються та змінюються об’єкти у відкритій та закритій областях пам’яті.

– Відбувається управління сертифікатами та USB-ключем.

У іншому випадку, тобто, якщо ПІН-код не дійсний, тоді виводиться повідомлення про недійсність введеного ПІН-коду.

Якщо ж користувач входить з правами гостьового доступу, тоді він може тільки переглядати об’єкти у відкритій області пам’яті.

Після виконання усіх вищезазначених етапів роботи програми користувач може покинути систему.

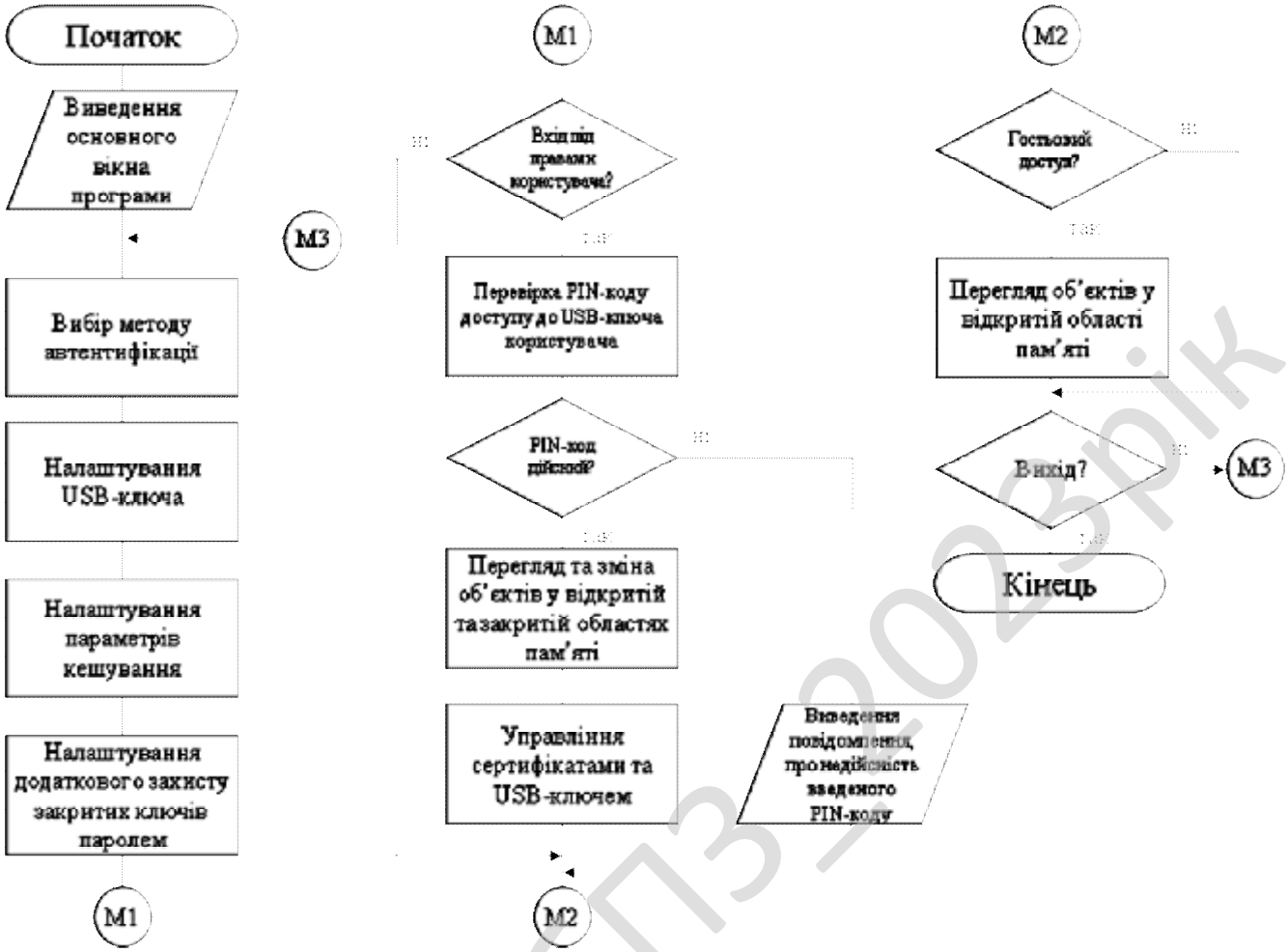

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

Як було сказано раніше система в момент реалізації процедури автентифікації працює наступним чином. На Хмарний сервіс встановлюється драйвер USB-ключа. Користувач під'єднує при вході у систему USB-ключ, для початку процедури автентифікації. Система видає запит ПІН-коду доступу до USB-ключа. Після введення ПІН-коду, видається вікно у якому потрібно ввести логін та пароль, необхідний для виконання процедури автентифікації й допуску користувача до системи. 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. Програмне забезпечення, встановлене на флеш-накопичувачі, згідно заданих таємних криптографічних алгоритмів перетворює цю частину ключа й надсилає відповідь.

У системі у якості таємничого криптографічного алгоритму використовується алгоритм RSA.

На рисунку 4.2 зображено блок-схему підпрограм шифрування та дешифрування за допомогою алгоритму RSA.

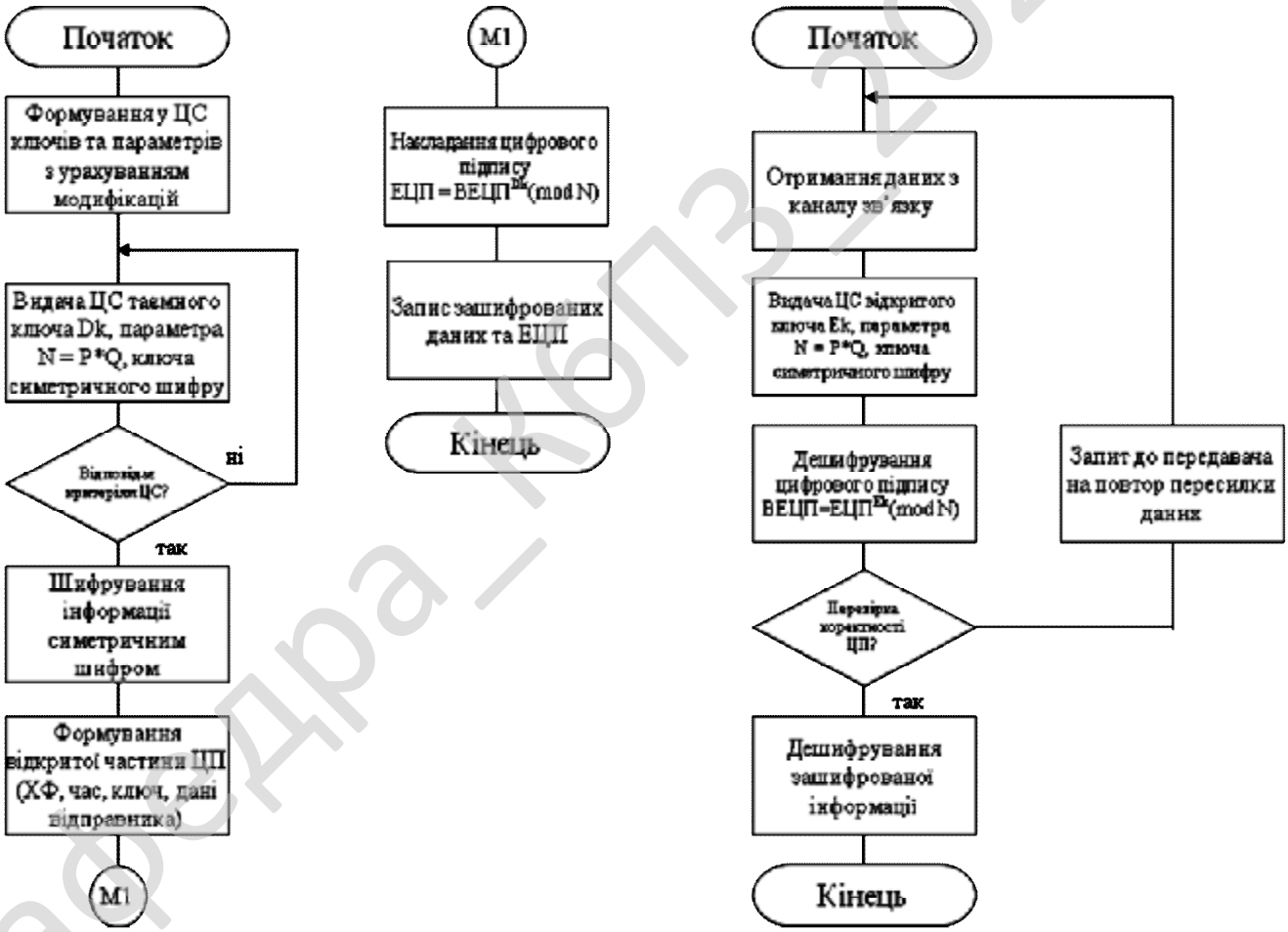

Рисунок 4.2 – Блок-схема підпрограм шифрування та дешифрування за допомогою алгоритму RSA

З блок схеми видно, що фактично алгоритм підпрограми розділяється на два алгоритми:

– Алгоритм шифрування методом RSA.

– Алгоритм дешифрування методом RSA.

Алгоритм шифрування методом RSA працює наступним чином:

– Відбувається формування у центрі сертифікації (ЦС) ключів та параметрів з урахуванням модифікації алгоритму.

– Відбувається видача центром сертифікації таємного ключа, параметра модуля шифрування, та ключа симетричного шифру.

– Якщо ці параметри не відповідають критеріям ЦС, тоді відбувається повторна видача, ще раз згенерованих інших параметрів. У іншому випадку програма продовжує свою роботу наступним чином.

– Відбувається шифрування інформації симетричним ключем. У якості алгоритму симетричного шифрування обрано алгоритм AES.

– Відбувається формування відкритої частини цифрового підпису, який включає в себе наступні дані: геш-фкнкцію, час, ключ, дані відправника.

– Відбувається накладання цифрового підпису, згідно формул визначених у алгоритмі шифрування методом RSA.

– Відбувається запис зашифрованих даних та електроного цифрового підпису (ЕЦП).

Алгоритм дешифрування методом RSA працює наступним чином:

– Відбувається отримання даних з каналу зв’язку, у даному випадку з USB-каналу.

– Відбувається видача центром сертифікації відкритого ключа, параметра модуля шифрування, ключас симетричного алгоритму.

– Відбувається дешифрування цифрового підпису.

– Якщо цифровий підпис є коректним, тоді відбувається дешифрування інформації, яка міститься на закритій частині носія інформації.

– Якщо цифровий підпис є некоректним, тді відбувається запит до

передавача на повтор пересилки даних, або забороняється доступ, якщо кількість запитів перевищила встановлену адміністратором системи.

– Після цього підпрограма дешифруваня закінчує свою роботу.

Щоб криптозахист не можна було «обійти» з іншого боку – приміром, перехопити з незахищеної області пам'яті секретні паролі – криптографічні функції повинні бути частиною операційної системи. У сімействі Windows, починаючи з Windows 95, забезпечується реалізація шифрування, генерації ключів, створення й перевірки цифрових підписів і інших криптографічних завдань. Ці функції необхідні для роботи операційної системи, однак ними може скористатися й будь-яка прикладна програма – для цього програмістові досить звернутися до потрібної підпрограми так, як пропонує криптографічний інтерфейс прикладних програм (CryptoAPI).

Зрозуміло, у міру вдосконалювання Windows розширювався й состав її криптографічної підсистеми. Крім базових операцій, у цей час в CryptoAPI 2.0 підтримується робота із сертифікатами, шифрованими повідомленнями у форматі PKCS #7 та ін. Опис функцій CryptoAPI, крім спеціальних книг, можна знайти в MSDN Library.

Взаємодія з CryptoAPI

Функції CryptoAPI можна викликати із програми, написаної мовою C++. Проте, Pascal де-факто визнаний стандартом в області навчання програмуванню. Крім того, у ряді вітчизняних компаній Delphi є базовим засобом розробки. Тому система була реалізована в середовищі Delphi. Хоча як інструмент можна було б вибрати й MS Visual C++.

Код функцій криптографічної підсистеми втримується в декількох бібліотеках, що завантажуються динамічно, Windows (advapi32.dll, crypt32.dll). Для звертання до такої функції із прикладної програми на Object Pascal варто оголосити її як зовнішню. Заголовок функції в інтерфейсної частини модуля буде виглядати, наприклад, так:

```
function CryptAcquireContext (
phPROV: PHCRYPTPROV;
```

```
pszContainer: LPCTSTR;
pszProvider: LPCTSTR;
dwProvType: DWORD;
dwFlags: DWORD): BOOL; stdcall;
```

а у частині, що виконується, замість тіла функції потрібно вписати директиву extern із вказівкою бібліотеки, у якій утримується функція, і, можливо, її ім'я в цій бібліотеці (якщо воно відрізняється від імені функції в створюваному модулі), наприклад:

```
function CryptAcquireContext; external 'advapi32.dll'
name 'CryptAcquireContext';
```

Таким чином, маючи опис функцій CryptoAPI, можна зібрати заголовки функцій в окремому модулі, що буде забезпечувати взаємодію прикладної програми із криптографічною підсистемою. Зрозуміло, така робота була проробленa програмістами Microsoft, і відповідний заголовний файл (wincrypt.h) був включений у поставку MS Visual C++. На щастя, з'явилася й Delphi-версія (wcrypt2.pas). Підключивши модуль до проекту, можливо використовувати не тільки функції CryptoAPI, але й мнемонічні константи режимів, ідентифікатори алгоритмів і інших параметрів, необхідних на практиці.

Ряд функцій був реалізований тільки в Windows 2000 і вище. Але й на Windows 98 можна знайти управу: при установці Internet Explorer 5 бібліотеки, що нас цікавлять, обновляються, дозволяючи використовувати новітні криптографічні можливості. Потрібно лише задати для Delphi-проекту параметр умовної компіляції NT5, після чого виклики функцій, що з'явилися лише в Windows 2000, будуть нормально працювати.

Знайомство із криптопровайдерами

Функції CryptoAPI забезпечують прикладним програмам доступ до криптографічних можливостей Windows. Однак вони є лише «передатною ланкою» у складному ланцюзі обробки інформації. Основну роботу виконують сховані від очей програміста функції, що входять у спеціалізовані програмні (або програмно-апаратні) модулі – провайдери (постачальники) криптографічних послуг (CSP – Cryptographic Service Providers), або криптопровайдери.

Програмна частина криптопровайдеру являє собою dll-файл, підписаний Microsoft; періодично Windows перевіряє цифровий підпис, що виключає можливість підміни криптопровайдеру.

Криптопровайдери відрізняються друг від друга:

– составом функцій (наприклад, деякі криптопровайдери не виконують шифрування даних, обмежуючись створенням і перевіркою цифрових підписів);

– вимогами до встаткування (спеціалізовані криптопровайдери можуть вимагати пристрої для роботи зі смарт-картами для виконання автентифікації користувача);

– алгоритмами, що здійснюють базові дії (створення ключів, гешування та ін.).

По составу функцій і їхніх алгоритмів, що забезпечують, криптопровайдери підрозділяються на типи. Наприклад, будь-який CSP типу PROV_RSA_FULL підтримує як шифрування, так і цифрові підписи, використовує для обміну ключами й створення підписів алгоритм RSA, для шифрування – алгоритми RC2 і RC4, а для гешування – MD5 і SHA.

Залежно від версії операційної системи состав установлених криптопровайдерів може істотно змінюватися. Однак на будь-якому комп'ютері з Windows можна знайти Microsoft Base Cryptographic Provider, що ставиться до уже відомому нам типу PROV_RSA_FULL. Саме із цим провайдером за замовчуванням будуть взаємодіяти всі програми.

Використання криптографічних можливостей Windows нагадує роботу програми із графічним пристроєм. Криптопровайдер подібний до графічного драйвера: він може забезпечувати взаємодію програмного забезпечення з устаткуванням (пристрій читання смарт-карт, апаратні датчики випадкових чисел та ін.). Для виводу інформації на графічний пристрій додаток не повинне безпосередньо звертатися до драйвера – замість цього потрібно одержати в системи контекст пристрою, за допомогою якого й здійснюються всі операції. Це дозволяє прикладному програмістові використовувати графічний пристрій,

нічого не знаючи про його апаратну реалізацію. Точно так само для використання криптографічних функцій додаток звертається до криптопровайдеру не прямо, а через CryptoAPI. При цьому спочатку необхідно запросити в системи контекст криптопровайдеру.

Першою справою, хоча б із цікавості, з'ясуємо, які ж криптопровайдери встановлені в системі. Для цього нам знадобляться чотири функції CryptoAPI:

– CryptEnumProviders (i, резерв, прапори, тип, ім'я, довжина_ім'я) – повертає ім'я й тип i-го один по одному криптопровайдеру в системі (нумерація починається з нуля);

– CryptAcquireContext (провайдер, контейнер, ім'я, тип, прапори) – виконує підключення до криптопровайдеру із заданим типом і ім'ям і повертає його дескриптор (контекст). При підключенні ми будемо передавати функції прапор CRYPT_VERIFYCONTEXT, службі для одержання контексту без підключення до контейнера ключів;

– CryptGetProvParam (провайдер, параметр, дані, розмір_даних, прапори) – повертає значення зазначеного параметра провайдеру, наприклад, версії (другий параметр при виклику функції – PP_VERSION), тип реалізації (програмний, апаратний, змішаний – PP_IMPTYPE), підтримувані алгоритми (PP_ENUMALGS). Список підтримуваних алгоритмів за допомогою цієї функції може бути отриманий у такий спосіб: при одному виклику функції вертається інформація про один алгоритм; при першому виклику функції варто передати значення прапора CRYPT_FIRST, а при наступний прапор повинен бути дорівнює 0;

– CryptReleaseContext (провайдер, прапори) – звільняє дескриптор криптопровайдеру.

Кожна із цих функцій, як і більшість інших функцій CryptoAPI, повертає логічне значення, рівне true, у випадку успішного завершення, і false – якщо виникли помилки. Код помилки може бути отриманий за допомогою функції GetLastError. Можливі значення кодів помилки наведені в згадуваній вище

документації. Наприклад, при виклику функції CryptGetProvParam для одержання версії провайдеру варто врахувати можливість виникнення помилок .

Процедура, що виводить в Memo-полі FileMemo викликається при виборі відповідного елемента в головному меню форми. Для стислості в тексті програми опущені фрагменти, що виконують обробку помилок.

Шифрування з використанням паролів

Після того як ми довідалися дещо про структуру CryptoAPI, можна скористатися нею в практичних цілях. Мабуть, самою очікуваною дією криптографічної підсистеми є шифрування файлів – так, щоб лише користувач, що знає певний пароль, міг одержати до них доступ.

Для шифрування даних в CryptoAPI застосовуються симетричні алгоритми. Симетричність означає, що для шифрування й розшифровки даних використовується той самий ключ, відомий як стороні, яка шифрує, так і стороні, що розшифровує. При цьому погано обраний ключ шифрування може дати супротивникові можливість зламати шифр. Тому однією з функцій криптографічної підсистеми повинна бути генерація «гарних» ключів або випадковим образом, або на підставі деякої інформації, надаваної користувачем, наприклад пароля.

У випадку створення ключа на підставі пароля повинне виконуватися наступна обов'язкова умова: при багаторазовому повторенні процедури генерації ключа на тому самому паролі повинні виходити ідентичні ключі. Ключ шифрування має, як правило, строго певну довжину, обумовлену використовуваним алгоритмом, а довжина пароля може бути довільної. Навіть інтуїтивно зрозуміло, що для однозначної генерації ключів потрібно привести різноманітні паролі до деякої єдиної форми. Це досягається за допомогою гешування.

Гешуванням (від англ. hash – розрізати, кришити, перемішувати) називається перетворення рядка довільної довжини в бітову послідовність

фіксованої довжини (геш-значення, або просто геш) із забезпеченням наступних умов:

– по геш-значенню неможливо відновити вихідне повідомлення;

– практично неможливо знайти ще один текст, що дає такий же геш, як і наперед задане повідомлення;

– практично неможливо знайти два різних тексти, що дають однакові геш-значення (такі ситуації називають колізіями).

При дотриманні наведених умов геш-значення служить компактним цифровим відбитком (дайджестом) повідомлення. Існує безліч алгоритмів гешування. CryptoAPI підтримує, наприклад, алгоритми MD5 (MD – Message Digest) і SHA (Secure Hash Algorithm).

Отже, щоб створити ключ шифрування на підставі пароля, нам потрібно спочатку одержати геш цього пароля. Для цього варто створити за допомогою CryptoAPI геш-об'єкт, скориставшись функцією CryptCreateHash (провайдер, ID_алгоритму, ключ, прапори, геш), яке потрібно передати дескриптор криптопровайдеру (отриманий за допомогою CryptAcquireContext) і ідентифікатор алгоритму гешування (інші параметри можуть бути нулями). У результаті ми одержимо дескриптор геш-объекта. Цей об'єкт можна уявити собі як чорний ящик, що приймає будь-які дані й «перемелює» їх, зберігаючи усередині себе лише геш-значення. Подати дані на вхід геш-об'єкта дозволяє функція CryptHashData (дескриптор, дані, розмір_даних, прапори).

Безпосереднє створення ключа виконує функція CryptDeriveKey (провайдер, ID_алгоритму, геш-об'єкт, прапори, ключ), що приймає геш-об'єкт у якості вихідних даних і будує підходящий ключ для алгоритму шифрування, заданого своїм ID. Результатом буде дескриптор ключа, якому можна використовувати для шифрування.

Варто звернути увагу, що при роботі з CryptoAPI ми увесь час маємо справу не із самими об'єктами або їхніми адресами, а з дескрипторами – цілими числами, що характеризують положення об'єкта у внутрішніх таблицях

криптопровайдеру. Самі таблиці розташовуються в захищеній області пам'яті, так що програми-«шпигуни» не можуть одержати до них доступ.

Алгоритми шифрування, підтримувані CryptoAPI, можна розділити на блокові і потокові: перші обробляють дані відносно великими за розміром блоками (наприклад, 64, 128 бітів або більше), а другі – побітно (теоретично, на практиці ж – побайтно). Якщо розмір даних, підлягаючих шифруванню, не кратний розміру блоку, то останній, неповний блок даних, буде доповнений необхідною кількістю випадкових бітів, у результаті чого розмір зашифрованої інформації може трохи збільшитися. Зрозуміло, при використанні потокових шифрів розмір даних при шифруванні залишається незмінним.

Шифрування виконується функцією CryptEncrypt (ключ, геш, фінал, прапори, дані, розмір_даних, розмір_буфера):

– через параметр ключ передається дескриптор ключа шифрування;

– параметр геш використовується, якщо одночасно із шифруванням потрібно обчислити геш-значення шифруємого тексту;

– параметр фінал дорівнює true, якщо шифруємий блок тексту – останнє або єдиний (шифрування можна здійснювати частинами, викликаючи функцію CryptEncrypt кілька разів);

– значення прапора повинне бути нульовим;

– параметр дані являє собою адреса буфера, у якому при виклику функції перебуває вихідний текст, а по завершенню роботи функції – зашифрований;

– наступний параметр, відповідно, описує розмір вхідних/вихідних даних,

– останній параметр задає розмір буфера – якщо в результаті шифрування зашифрований текст не вміститься в буфері, виникне помилка.

Для розшифровки даних використовується функція CryptDecrypt (ключ, геш, фінал, прапори, дані, розмір_даних), що відрізняється від функції, що шифрує, тільки тим, що розмір буфера вказувати не слід: оскільки розмір даних при розшифровці може тільки зменшитися, відведеного під них буфера напевно буде досить.

Звичайно, шифрування всіх файлів тим самим паролем полегшує «супротивникові» завдання їхньої розшифровки, запам'ятовування величезного числа паролів сильно ускладнює життя, а їхнє записування в незашифрованому вигляді створює небезпека розкриття всієї системи. CryptoAPI може запропонувати на цей випадок ряд рішень.

В арсеналі захисту повинні бути не тільки методи, що забезпечують таємність передачі інформації (про їх ми говорили в першій частині статті). Не менш важливими інструментами безпеки є процедури, що дозволяють переконатися в цілості й автентичності даних. Крім того, необхідно вирішувати проблеми безпечного зберігання й розподілу ключів.

Проблема розподілу ключів

За допомогою CryptoAPI ми вирішили таку "класичну" завдання як шифрування на основі пароля. Нагадаємо, що пароль використовувався для створення ключа шифрування якого-небудь симетричного алгоритму. У такому випадку розшифрувати файл може лише той, хто знає пароль. А виходить, для забезпечення конфіденційності потрібно тримати пароль у найсуворішому секреті – бажано, щоб його знали лише відправник і одержувач інформації. (А ще краще, якщо відправник і одержувач – одна й та сама особа).

Припустимо, що відправник і одержувач при особистій зустрічі домовилися використовувати для конфіденційної переписки певний пароль. Але якщо вони будуть шифрувати всі свої повідомлення тим самим ключем, те можливий супротивник, перехопивши кореспонденцію, буде мати гарні шанси зламати шифр: при сучасних методах криптоаналізу наявність декількох шифртекстів, отриманих шляхом використання того самого ключа, майже гарантує успішний результат. Тому при використанні симетричних алгоритмів шифрування настійно рекомендується не застосовувати той самий ключ двічі.

Однак пам'ятати окремий пароль для кожного зашифрованого повідомлення – завдання досить трудомістка. А для кореспондентів, не спроможних зустрітися особисто для узгодження ключів шифрування,

конфіденційний обмін повідомленнями взагалі стає недоступним. Такі практичні труднощі називаються проблемою розподілу ключів.

Рятівний спосіб, що дозволяє шифрувати повідомлення, обмінюючись ключами по відкритих каналах зв'язку, був придуманий у середині 70-х років минулого сторіччя, а на початку вісімдесятих з'явився перший його алгоритм, що реалізує, – RSA. Тепер користувач може згенерувати два зв'язаних між собою ключа – ключову пару. Один із цих ключів по несекретних каналах розсилається всім, з ким користувач хотів би обмінюватися конфіденційними повідомленнями. Цей ключ називають відкритим (англ. public key). Знаючи відкритий ключ користувача, можна зашифрувати адресоване йому повідомлення, але от розшифрувати його дозволяє лише друга частина ключової пари – закритий ключ (private key). При цьому відкритий ключ не дає "практичний" можливості обчислити закритий: таке завдання, хоч і розв'язне в принципі, але при досить великому розмірі ключа вимагає багатьох років машинного часу. Для збереження конфіденційності одержувачеві необхідно лише зберігати в строгому секреті свій закритий ключ, а відправникові – переконатися, що наявний у нього відкритий ключ дійсно належить адресатові.

Так як для шифрування й розшифровки використовуються різні ключі, алгоритми такого роду назвали асиметричними. Найбільш істотним їхнім недоліком є низька продуктивність – вони приблизно в 100 разів повільніші симетричних алгоритмів. Тому були створені криптографічні схеми, що використовують переваги як симетричних, так і асиметричних алгоритмів:

– для шифрування файлу або повідомлення використовується швидкий симетричний алгоритм, причому ключ шифрування генерується випадковим образом із забезпеченням "гарних" статистичних властивостей;

– невеликий по розмірах симетричний ключ шифрування шифрується за допомогою асиметричного алгоритму з використанням відкритого ключа адресата й у зашифрованому вигляді пересилається разом з повідомленням;

– одержавши повідомлення, адресат своїм закритим ключем розшифровує симетричний ключ, а з його допомогою – і саме повідомлення.

Описана схема реалізована й в CryptoAPI.

Цілісність і автентичність інформації

Як упевнитися в тому, що повідомлення, що прийшло, дійсно відправлене тим, чиє ім'я коштує в графі "відправник"? Асиметричні схеми шифрування дають нам елегантний спосіб автентифікації. Якщо відправник зашифрує повідомлення своїм закритим ключем, то успішне розшифровування переконає одержувача в тому, що послати кореспонденцію міг тільки хазяїн ключової пари, і ніхто інший. При цьому розшифровку може виконати кожної, хто має відкритий ключ відправника. Адже наша мета – не конфіденційність, а автентифікація.

Щоб уникнути шифрування всього повідомлення за допомогою асиметричних алгоритмів, використовують гешування: обчислюється геш-значення вихідного повідомлення, і тільки ця коротка послідовність байтів шифрується закритим ключем відправника. Результат являє собою електронний цифровий підпис. Додавання такого підпису до повідомлення дозволяє встановити:

– автентичність повідомлення – створити підпис на основі закритого ключа міг тільки його хазяїн;

– цілісність даних – легко обчислити геш-значення отриманого повідомлення й зрівняти його з тим, що зберігається в підписі: якщо значення збігаються, виходить, повідомлення не було змінено зловмисником після того, як відправник його підписав.

Таким чином, асиметричні алгоритми дозволяють вирішити два непрості завдання: обміну ключами шифрування по відкритих каналах зв'язку й підпису повідомлення. Щоб скористатися цими можливостями, потрібно згенерувати й зберегти дві ключові пари – для обміну ключами й для підписів. У цьому нам допоможе CryptoAPI.

Контейнери ключів

Кожний криптопровайдер має у своєму розпорядженні базу даних, у якій зберігаються довгострокові ключі користувачів. База даних містить один або більше контейнерів ключів. Користувач може створити кілька контейнерів з різними іменами (ім'ям контейнера за замовчуванням є ім'я користувача в системі).

Підключення до контейнера виробляється одночасно з одержанням контексту криптопровайдеру при виклику функції CryptAcquireContext – ім'я контейнера ключів передається функції другим її аргументом. Якщо другий аргумент містить порожній покажчик (nil), то використовується ім'я за замовчуванням, тобто ім'я користувача. У тому випадку, якщо доступ до контейнера не потрібний, можна передати в останньому аргументі функції прапор CRYPT_VERIFYCONTEXT; при необхідності створити новий контейнер використовується прапор CRYPT_NEWKEYSET; а для видалення існуючого контейнера разом із ключами, що зберігаються в ньому, – CRYPT_DELETEKEYSET.

Кожний контейнер може містити, як мінімум, дві ключові пари – ключ обміну ключами й ключ підпису. Ключі, використовувані для шифрування симетричними алгоритмами, не зберігаються. Як ми вже говорили, такі ключі не рекомендується застосовувати більше одного разу, тому їх називають сеансовими (англ. session key).

Створення ключових пар

Після створення контейнера ключів необхідно згенерувати ключові пари обміну ключами й підпису. Цю роботу в CryptoAPI виконує функція CryptGenKey (провайдер, алгоритм, прапори, ключ):

– провайдер – дескриптор криптопровайдеру, отриманий у результаті звертання до функції CryptAcquireContext;

– алгоритм – указує, якому алгоритму шифрування буде відповідати створюваний ключ. Інформація про алгоритм, таким чином, є частиною опису ключа. Кожний криптопровайдер використовує для обміну ключами й підпису

строго певні алгоритми. Так, провайдери типу PROV_RSA_FULL, до яких ставиться й Microsoft Base Cryptographic Provider, реалізують алгоритм RSA. Але при генерації ключів знати це не обов'язково: досить указати, який ключ ми збираємося створити – обміну ключами або підпису. Для цього використовуються мнемонічні константи AT_KEYEXCHANGE і AT_SIGNATURE;

– прапори – при створенні асиметричних ключів управляє їхнім розміром. Використовуваний нами криптопровайдер дозволяє генерувати ключ обміну ключами довжиною від 384 до 512 біт**, а ключ підпису – від 512 до 16384 біт. Чим більше довжина ключа, тим вище його надійність, тому важко знайти причини для використання ключа обміну ключами довжиною менш 512 біт, а довжину ключа підпису не рекомендується робити менше 1024 біт**. За замовчуванням криптопровайдер створює обидва ключі довжиною 512 біт. Необхідну довжину ключа можна передати в старшому слові параметра прапори;

– ключ – у випадку успішного завершення функції в цей параметр заноситься дескриптор створеного ключа.

Розглянемо приклад створення ключових пар. У полі "Контейнер" можна вказати ім'я контейнера ключів; якщо залишити це поле порожнім, буде використаний контейнер за замовчуванням. Призначення інших елементів керування повинне бути інтуїтивно зрозумілим. Після генерації ключа в memo-полі виводиться звіт про його параметри. Для цього використовується функція CryptGetKeyParam (ключ, параметр, буфер, розмір, прапори). Щоб одержати інформацію про необхідний параметр, потрібно через другий аргумент функції передати відповідну константу: KP_ALGID – ідентифікатор алгоритму, KP_KEYLEN – розмір ключа, і т.д.

Обмін ключами

Тепер маємо набір ключів, однак всі вони не потрібні, доти поки ми не одержимо можливості обміну з іншими користувачами відкритими ключами. Для цього необхідно витягти їх з бази даних ключів і записати у файл, який можна

буде передати своїм кореспондентам. При експорті дані ключа зберігаються в одному із трьох можливих форматів:

– PUBLICKEYBLOB – використовується для збереження відкритих ключів. Оскільки відкриті ключі не є секретними, вони зберігаються в незашифрованому вигляді;

– PRIVATEKEYBLOB – використовується для збереження ключової пари цілком (відкритого й закритого ключів). Ці дані є найвищою мірою секретними, тому зберігаються в зашифрованому вигляді, причому для шифрування використовується сеансовий ключ (і, відповідно, симетричний алгоритм);

– SIMPLEBLOB – використовується для збереження сеансових ключів. Для забезпечення таємності дані ключа шифруються з використанням відкритого ключа одержувача повідомлення.

Експорт ключів в CryptoAPI виконується функцією CryptExportKey (експортований ключ, ключ адресата, формат, прапори, буфер, розмір буфера):

– експортований ключ – дескриптор потрібного ключа;

– ключ адресата – у випадку збереження відкритого ключа повинен бути дорівнює нулю (дані не шифруються);

– формат – вказується один з можливих форматів експорту (PUBLICKEYBLOB, PRIVATEKEYBLOB, SIMPLEBLOB);

– прапори – зарезервований на майбутнє (повинен бути дорівнює нулю);

– буфер – містить адреса буфера, у який буде записаний ключовий BLOB (Binary Large OBject – великий двійковий об'єкт);

– розмір буфера – при виклику функції в цій змінній повинен перебувати доступний розмір буфера, а по закінченні роботи в неї записується кількість експортованих даних. Якщо розмір буфера заздалегідь не відомий, то функцію потрібно викликати з параметром буфер, рівним порожньому покажчику, тоді розмір буфери буде обчислений і занесений у змінну розмір буфера.

Експорт ключової пари цілком, включаючи й закритий ключ, може знадобитися для того, щоб мати можливість підписувати документи на різних комп'ютерах (наприклад, удома й на роботі), або для збереження страховочної

копії. У цьому випадку потрібно створити ключ шифрування на підставі пароля й передати дескриптор цього ключа як другий параметр функції CryptExportKey.

Запросити в криптопровайдеру дескриптор самого експортованого ключа дозволяє функція CryptGetUserKey (провайдер, опис ключа, дескриптор ключа). Опис ключа – це або AT_KEYEXCHANGE, або AT_SIGNATURE.

Експортовані в такий спосіб відкриті частини ключів знадобляться нам для перевірки підпису й розшифровки сеансового ключа.

Імпорт ключових пар у знову створений контейнер – це самостійна процедура. Необхідно запросити в користувача назва контейнера й пароль, підключитися до провайдеру, створити на підставі пароля ключ, уважати з файлу імпортовані дані в буфер, після чого скористатися функцією CryptImportKey (провайдер, буфер, довжина буфера, ключ для розшифровки, прапори, імпортований ключ). Якщо потрібно забезпечити можливість експорту імпортованої ключової пари згодом, то в параметрі прапори необхідно передати значення CRYPT_EXPORTABLE; у противному випадку виклик для даної ключової пари функції CryptExportKey приведе до помилки.

Ми вже обговорювали, що при роботі з асиметричними алгоритмами важливо переконатися, що відкритий ключ дійсно належить тому, кого ви вважаєте його хазяїном, і не був підмінений зловмисником. Найпростішим способом забезпечити автентичність ключа є побайтне звірення з оригіналом, що зберігається в його хазяїна. Для цього можна просто дозволити користувачам переглянути експортовані дані в шістнадцятковому вигляді – наприклад, відкрити файл, у який був записаний відкритий ключ, і вивести його вміст у вікно перегляду.

Електронний цифровий підпис

Для створення електронного цифрового підпису необхідно обчислити геш заданого файлу й зашифрувати цей "цифровий відбиток повідомлення" своїм закритим ключем – "підписати". Щоб підпис згодом можна був перевірити, необхідно вказати, який алгоритм гешування використовувався при її створенні.

Підписати обчислений геш в CryptoAPI дозволяє функція CryptSignHash (геш, опис ключа, коментар, прапори, підпис, довжина підпису). Другим параметром може бути або AT_KEYEXCHANGE, або AT_SIGNATURE (у нашім випадку логічніше використовувати ключ підпису). Третій параметр із метою безпеки настійно рекомендується залишати порожнім (nil). Прапори в цей час також не використовуються – на місці цього аргументу повинен бути нуль. Готовий електронний підпис функція запише в буфер, адресу якого втримується в передостанньому параметрі, останній же параметр буде містити довжину підпису в байтах.

Щоб перевірити правильність підпису, одержувач підписаного повідомлення повинен мати файл із відкритим ключем підпису відправника. У процесі перевірки підпису цей ключ імпортується усередину криптопровайдеру. Перевірка виконується функцією CryptVerifySignature (геш, підпис, довжина підпису, відкритий ключ, коментар, прапори). Про останні два аргументи можна сказати те ж, що й про параметри коментар і прапори функції CryptSignHash, призначення ж інших повинне бути зрозуміло. Якщо підпис вірний, функція повертає true. Значення false як результат може свідчити або про виникнення помилки в процесі перевірки, або про те, що підпис виявився невірною. В останньому випадку функція GetLastError поверне помилку NTE_BAD_SIGNATURE.

Для конфіденційного обміну інформацією з кореспондентом у будь-якій крапці земної кулі доводиться використовувати цілий арсенал сучасних криптографічних інструментів: симетричні й асиметричні алгоритми шифрування, механізми генерування криптографічних ключів і випадкових послідовностей, специфічні режими роботи шифрів та ін. Продовжуючи тему, почату в першій і другій частинах статті, розглянемо реалізацію цих інструментів в CryptoAPI і скористаємося ними для шифрування файлу випадковим ключем.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм RC5, який являє собою блоковий шифр із безліччю параметрів: розміром блоку, розміром ключа й числом раундів. В алгоритмі RC5 передбачені три операції: XOR, додавання й циклічні зрушення. На більшості процесорів операції циклічного зрушення виконуються за постійний час, змінні циклічні зрушення являють собою нелінійну функцію. Циклічні зрушення залежать як від ключа, так і від даних.

В RC5 використовується блок змінної довжини, але в приводиться прикладі, що буде розглянутий, 64-бітовий блок даних. Шифрування використовує $2r+2$ залежних від ключа 32-бітових слів – S_0, S_1, S_2,... S_{2r+1} – де r – число раундів. Для шифрування спочатку потрібно розділити блок відкритого тексту на два 32-бітових слова: A и B. (При впакуванні байтів у слова в алгоритмі RC5 дотримується угода про прямий порядок (little-endian) байтів: перший байт займає молодші біти регістра A й т. ін.) Потім:

$$A=A + S_0$$

$$B = B + S_0$$

Для і від 1 до r:

$$A = ((A\oplus \quad B) <<< B) + S_{2i}$$

$$B = ((B\oplus \quad A) <<< A) + S_{2i+1}$$

Вихід перебуває в регістрах A и B.

Розшифрування теж нескладно. Потрібно розбити блок відкритого тексту на два слова, A й B, а потім:

Для i від r до 1 із кроком -1:

$$B = ((B - S_{2i+1}) >>> A) \oplus A$$

$$A = ((A - S_{2i}) >>> B) \oplus B$$

$$B = B - S_i$$

$$A = A - S_0$$

Символом «>>>» позначене циклічне зрушення вправо**.** Звичайно ж**,** всі додавання й вирахування виконуються по модулю 2^{32}.

Створення масиву ключів складніше, але теж прямолінійно. Спочатку байти ключа копіюються в масив L із 32-бітових слів, доповнюючи при необхідності заключне слово нулями. Потім масив S ініціалізується за допомогою лінійного конгруентного генератора по модулю 2^{32}:

$$S_0 = P$$

Для i від 1 до $2(r + 1) – 1$:

$$S_i = (S_{i-1} + Q) \bmod 2^{32}$$

де P = 0xb7e15163 і Q = 0x9e3779b9.

Нарешті, потрібно підставити L в S:

$$i = j = 0$$

$$A = B = 0$$

Виконати $3n$ раз (де n – максимум від $2(r + 1)$ і c):

$$A = S_i = (S_i + A + B) <<< 3$$

$$B = L_i = (L_i + A + B) <<< (A + B)$$

$$i = (i + 1) \bmod 2(r +1)$$

$$i = (j +1) \bmod c$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс головного вікна програми. З рисунку видно, що головне вікно складається з наступних блоків:

– Блок меню.

– Блок клавіш швидкого доступу до функцій програми.

– Вікно розподілу паролів та прав доступу, для різних користувачів системи.

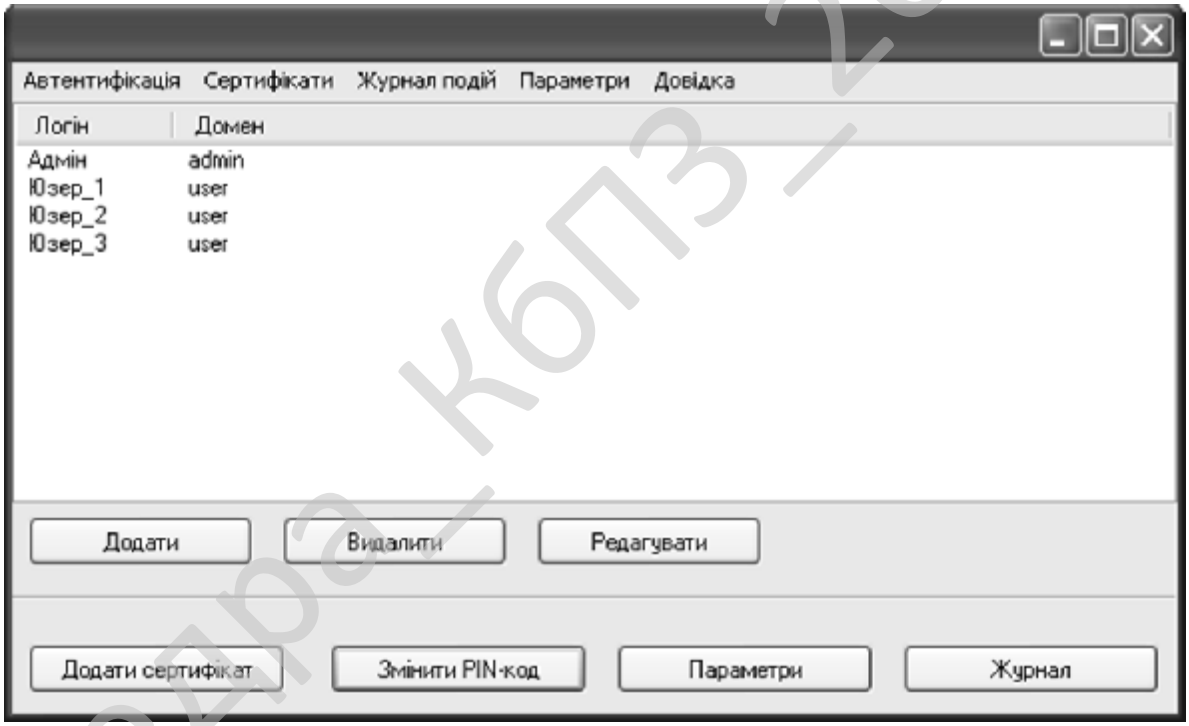

Рисунок 5.1 – Головне вікно програми

Блок меню складається з наступних елементів:

– Автентифікація.

– Сертифікати.

– Журнал подій.

– Параметри.

– Довідка.

Блок клавіш швидкого доступу до функцій програми складається з наступних клавіш:

– Додати сертифікат.

– Змінити ПІН-код.

– Параметри.

– Журнал.

У вікні розподілу паролів та прав доступу, для різних користувачів системи можливо додати, або видалити користувачів, або змінити права доступу, пароль, або ідентифікатор користувача.

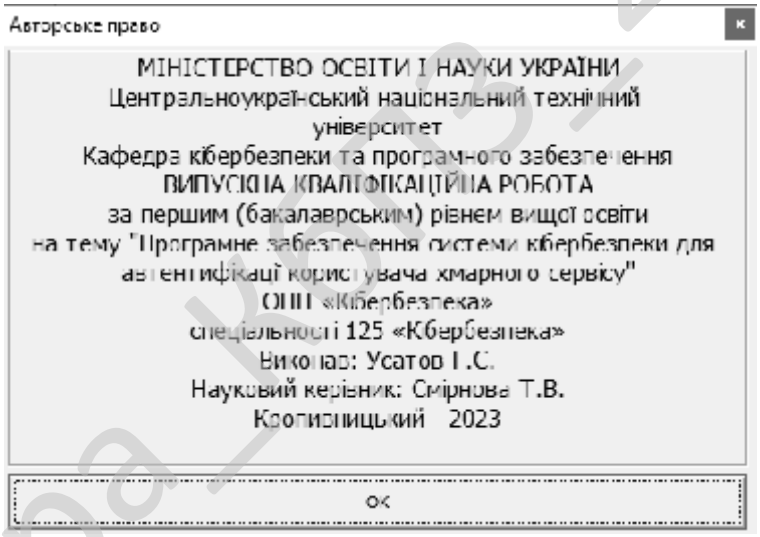

Рисунок 5.2 – Довідка

На рисунку 5.2 зображене вікно довідки, з якого видна інформація про розробника системи, його керівника та місце виконання бакалаврського проектування.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для автентифікації користувача хмарного сервісу.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для автентифікації користувача хмарного сервісу.

– Досліджена система для автентифікації користувача хмарного сервісу.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для автентифікації користувача хмарного сервісу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для автентифікації користувача хмарного сервісу.

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 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки для автентифікації користувача хмарного сервісу. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм RC5.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

2. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

3. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

4. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

5. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

6. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях

/ Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. – Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

7. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

8. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

9. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

10. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

11. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

12. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

13. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

14. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

15. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

16. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

17. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

18. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

19. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани,

С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

20. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

21. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

22. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

23. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

24. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та

інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

25. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

26. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

27. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

28. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

29. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

30. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

31. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

32. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

33. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

34. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

35. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов,

С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

36. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

37. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

38. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

39. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

40. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

41. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

42. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

43. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

44. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

45. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

46. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

47. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.

48. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing // IEEE/ACM Transactions on Networking. – 2001. – №9(2), P. 162-176.

49. Basabi Chakraborty. Simultaneous Search for Multiple Routes using Genetic Algorithm / IEEE lntemational Conference on Computational oltelligencs for Measurement System and Applications Boston. MA, USA, 14-16, July 2004, P. 77-80/

50. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: a survey // IEEE Communications Magazine. – 2000. – №38(1). – P. 40 - 46.

51. Carpenter G., A., Grossberg S. Pattern Recognition by Self-Organizing Neural Networks, Cambridge, MA, MIT Press, 1991.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для автентифікації користувача хмарного сервісу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 12-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для автентифікації користувача хмарного сервісу;

– 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 Sydney.

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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 78 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки для автентифікації користувача хмарного сервісу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2023 року

Файл Authentication_RSA.pas – алгоритм шифрування RSA

```
Unit Authentication_RSA;

Interface

Uses Windows, SysUtils, Controls, Authentication_;

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


Implementation


{$H+}

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

Procedure RSAEncrypt(P : String; Var exp, modb : TAuthentication_; Var E :
String);
Var
   i, j, modbits : longint;
   PGInt, temp, zero : TAuthentication_;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToAuthentication_('0', zero);
   Authentication_ToBase2String(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);
      Base2StringToAuthentication_(tempstr3, PGInt);
      delete(tempstr1, 1, modbits - 1);
      If tempstr3 = '0' Then Authentication_Copy(zero, temp) Else
Authentication_MontgomeryModExp(PGInt, exp, modb, temp);
      Authentication_Destroy(PGInt);
      tempstr3 := '';
      Authentication_ToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod modbits) <> 0 Do tempstr3 := '0' + tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      Authentication_destroy(temp);
   End;

   While (tempstr2[1] = '0') And (length(tempstr2) > 1) Do delete(tempstr2, 1,
1);
   ConvertBase2To256(tempstr2, E);
   Authentication_Destroy(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 :
TAuthentication_; Var D : String);
Var
   i, j, modbits : longint;
   EGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp, zero : TAuthentication_;
   tempstr1, tempstr2, tempstr3 : String;
Begin
   Base2StringToAuthentication_('0', zero);
   Authentication_ToBase2String(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
      Authentication_ModInv(q, p, temp1);
      Authentication_Mul(q, temp1, qqinvp);
      Authentication_Destroy(temp1);
      Authentication_ModInv(p, q, temp1);
      Authentication_Mul(p, temp1, ppinvq);
      Authentication_Destroy(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);
      Base2StringToAuthentication_(tempstr3, EGInt);
      delete(tempstr1, 1, modbits);
      If tempstr3 = '0' Then Authentication_Copy(zero, temp) Else
      Begin
         If exp.Number <> Nil Then Authentication_MontgomeryModExp(EGInt, exp,
modb, temp) Else
         Begin
            Authentication_MontgomeryModExp(EGInt, d_p, p, temp1);
            Authentication_Mul(temp1, qqinvp, temp3);
            Authentication_Copy(temp3, temp1);
            Authentication_MontgomeryModExp(EGInt, d_q, q, temp2);
            Authentication_Mul(temp2, ppinvq, temp3);
            Authentication_Copy(temp3, temp2);
            Authentication_AddMod(temp1, temp2, modb, temp);
            Authentication_Destroy(temp1);
            Authentication_Destroy(temp2);
         End;
      End;
      Authentication_Destroy(EGInt);
      tempstr3 := '';
      Authentication_ToBase2String(temp, tempstr3);
      While (length(tempstr3) Mod (modbits - 1)) <> 0 Do tempstr3 := '0' +
tempstr3;
      tempstr2 := tempstr2 + tempstr3;
      Authentication_destroy(temp);
   End;

   If exp.Number = Nil Then
   Begin
      Authentication_Destroy(ppinvq);
      Authentication_Destroy(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);
   Authentication_Destroy(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 : TAuthentication_; Var S :
String);
Var
   MGInt, SGInt, temp, temp1, temp2, temp3, ppinvq, qqinvp : TAuthentication_;
Begin
   Base256StringToAuthentication_(M, MGInt);
   If d.Number <> Nil Then Authentication_MontgomeryModExp(MGInt, d, n, SGInt)
Else
   Begin
      Authentication_ModInv(p, q, temp);
      Authentication_Mul(p, temp, ppinvq);
      Authentication_Destroy(temp);
      Authentication_ModInv(q, p, temp);
      Authentication_Mul(q, temp, qqinvp);
      Authentication_Destroy(temp);
      Authentication_MontgomeryModExp(MGInt, dp, p, temp1);
      Authentication_Mul(temp1, qqinvp, temp2);
      Authentication_Copy(temp2, temp1);
      Authentication_MontgomeryModExp(MGInt, dq, q, temp2);
      Authentication_Mul(temp2, ppinvq, temp3);
      Authentication_Copy(temp3, temp2);
      Authentication_AddMod(temp1, temp2, n, SGInt);
      Authentication_Destroy(temp1);
      Authentication_Destroy(temp2);
      Authentication_Destroy(ppinvq);
      Authentication_Destroy(qqinvp);
   End;
   Authentication_ToBase256String(SGInt, S);
   Authentication_Destroy(MGInt);
   Authentication_Destroy(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 : TAuthentication_; Var valid :
boolean);
Var
   MGInt, SGInt, temp : TAuthentication_;
Begin
   Base256StringToAuthentication_(S, SGInt);
   Base256StringToAuthentication_(M, MGInt);
   Authentication_Mod(MGInt, n, temp);
   Authentication_Copy(temp, MGInt);
   Authentication_MontgomeryModExp(SGInt, e, n, temp);
   Authentication_Copy(temp, SGInt);
   valid := (Authentication_CompareAbs(SGInt, MGInt) = Eq);
   Authentication_Destroy(SGInt);
   Authentication_Destroy(MGInt);
End;

End.
```

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

```
Unit Authentication_PrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, Authentication_;

Procedure Authentication_Add(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Sum : TAuthentication_);
Procedure Authentication_Copy(Const Authentication_1 : TAuthentication_; Var
Authentication_2 : TAuthentication_);
Procedure Authentication_TrialDiv9999(Const Authentication_ : TAuthentication_;
Var ok : boolean);
Procedure Authentication_Primetest(Var Authentication_p : TAuthentication_;
nrRMtests : integer; Var ok : boolean);
Procedure Authentication_Destroy(Var Authentication_ : TAuthentication_);
Procedure PrimeSearch(Var GInt : TAuthentication_);

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 Authentication_Add(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Sum : TAuthentication_);
Var
   i, size1, size2, size : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.Number[0];
   If size1 < size2 Then Authentication_Add(Authentication_2, Authentication_1,
Sum) Else
   Begin
      If Authentication_1.Sign = Authentication_2.Sign Then
      Begin
         Sum.Sign := Authentication_1.Sign;
         SetLength(Sum.Number, size1 + 2);
         rest := 0;
         For i := 1 To size2 Do
         Begin
            Trest := Authentication_1.Number[i] + Authentication_2.Number[i] +
rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
         For i := (size2 + 1) To size1 Do
         Begin
            Trest := Authentication_1.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 Authentication_CompareAbs(Authentication_2, Authentication_1) = Lt
Then Authentication_Add(Authentication_2, Authentication_1, Sum)
         Else
         Begin
            SetLength(Sum.Number, size1 + 1);
            rest := 0;
            For i := 1 To size2 Do
            Begin
               Trest := 2147483648 + Authentication_1.Number[i] -
Authentication_2.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 + Authentication_1.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 := Authentication_1.Sign;
         End;
      End;
   End;
End;

Procedure Authentication_Copy(Const Authentication_1 : TAuthentication_; Var
Authentication_2 : TAuthentication_);
Begin
   Authentication_2.Sign := Authentication_1.Sign;
   Authentication_2.Number := Nil;
   Authentication_2.Number := Copy(Authentication_1.Number, 0,
Authentication_1.Number[0] + 1);
End;

Procedure Authentication_TrialDiv9999(Const Authentication_ : TAuthentication_;
Var ok : boolean);
Var
   j : int64;
   i : integer;
Begin
   If ((Authentication_.Number[1] Mod 2) = 0) Then ok := false
   Else
   Begin
      i := 0;
      ok := true;
      While ok And (i < 1228) Do
      Begin
         i := i + 1;
         Authentication_modbyint(Authentication_, primes[i], j);
         If j = 0 Then ok := false;
      End;
   End;
End;
// Перевірка на простоту числа
Procedure Authentication_Primetest(Var Authentication_p : TAuthentication_;
nrRMtests : integer; Var ok : boolean);
Begin
   Authentication_Trialdiv9999(Authentication_p, ok);
   If ok Then Authentication_RabinMiller(Authentication_p, nrRMtests, ok);
End;

Procedure Authentication_Destroy(Var Authentication_ : TAuthentication_);
Begin
   Authentication_.Number := Nil;
End;
//Пошук простих чисел
Procedure PrimeSearch(Var GInt : TAuthentication_);
Var
   temp, two : TAuthentication_;
   ok : Boolean;
Begin
   If (GInt.Number[1] Mod 2) = 0 Then GInt.Number[1] := GInt.Number[1] + 1;
   Base10StringToAuthentication_('2', two);
   ok := false;
   While Not ok Do
   Begin
```

```
      Authentication_Add(GInt, two, temp);
      Authentication_Copy(temp, GInt);
      Authentication_PrimeTest(GInt, 4, ok);
   End;
   Authentication_Destroy(two);
End;
End.
```

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

```
Unit Authentication_;

Interface

Uses Windows, SysUtils, Controls, Math;

Type
   TCompare = (Lt, St, Eq, Er);
   TSign = (negative, positive);
   TAuthentication_ = 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 Base10StringToAuthentication_(Base10 : String; Var Authentication_ :
TAuthentication_);
Procedure Authentication_ToBase10String(Const Authentication_ :
TAuthentication_; Var Base10 : String);
Procedure Authentication_Destroy(Var Authentication_ : TAuthentication_);
Function Authentication_CompareAbs(Const Authentication_1, Authentication_2 :
TAuthentication_) : TCompare;
Procedure Authentication_Add(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Sum : TAuthentication_);
Procedure Authentication_ChangeSign(Var Authentication_ : TAuthentication_);
Procedure Authentication_Sub(Var Authentication_1, Authentication_2, dif :
TAuthentication_);
Procedure Authentication_MulByInt(Const Authentication_ : TAuthentication_; Var
res : TAuthentication_; by : int64);
Procedure Authentication_MulByIntbis(Var Authentication_ : TAuthentication_; by
: int64);
Procedure Authentication_DivByInt(Const Authentication_ : TAuthentication_; Var
res : TAuthentication_; by : int64; Var modres : int64);
Procedure Authentication_DivByIntBis(Var Authentication_ : TAuthentication_; by
: int64; Var modres : int64);
Procedure Authentication_ModByInt(Const Authentication_ : TAuthentication_; by :
int64; Var modres : int64);
Procedure Authentication_Abs(Var Authentication_ : TAuthentication_);
Procedure Authentication_Copy(Const Authentication_1 : TAuthentication_; Var
Authentication_2 : TAuthentication_);
Procedure Authentication_ShiftLeft(Var Authentication_ : TAuthentication_);
Procedure Authentication_ShiftRight(Var Authentication_ : TAuthentication_);
Procedure Authentication_ShiftRightBy31(Var Authentication_ : TAuthentication_);
Procedure Authentication_AddBis(Var Authentication_1 : TAuthentication_; Const
Authentication_2 : TAuthentication_);
Procedure Authentication_SubBis(Var Authentication_1 : TAuthentication_; Const
Authentication_2 : TAuthentication_);
Procedure Authentication_Mul(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Prod : TAuthentication_);
```

```
Procedure Authentication_Square(Const Authentication_ : TAuthentication_; Var
Square : TAuthentication_);
Procedure Authentication_ToBase2String(Const Authentication_ : TAuthentication_;
Var S : String);
Procedure Base2StringToAuthentication_(S : String; Var Authentication_ :
TAuthentication_);
Procedure Authentication_ToBase256String(Const Authentication_ :
TAuthentication_; Var str256 : String);
Procedure Base256StringToAuthentication_(str256 : String; Var Authentication_ :
TAuthentication_);
Procedure PGPMPIToAuthentication_(PGPMPI : String; Var Authentication_ :
TAuthentication_);
Procedure Authentication_ToPGPMPI(Authentication_ : TAuthentication_; Var PGPMPI
: String);
Procedure Authentication_Exp(Const Authentication_, exp : TAuthentication_; Var
res : TAuthentication_);
Procedure Authentication_Fac(Const Authentication_ : TAuthentication_; Var res :
TAuthentication_);
Procedure Authentication_ShiftLeftBy31(Var Authentication_ : TAuthentication_);
Procedure Authentication_DivMod(Var Authentication_1, Authentication_2,
QAuthentication_, MAuthentication_ : TAuthentication_);
Procedure Authentication_Div(Var Authentication_1, Authentication_2,
QAuthentication_ : TAuthentication_);
Procedure Authentication_Mod(Var Authentication_1, Authentication_2,
MAuthentication_ : TAuthentication_);
Procedure Authentication_SquareMod(Var Authentication_, Modb, Authentication_SM
: TAuthentication_);
Procedure Authentication_AddMod(Var Authentication_1, Authentication_2, base,
Authentication_res : TAuthentication_);
Procedure Authentication_MulMod(Var Authentication_1, Authentication_2, base,
Authentication_res : TAuthentication_);
Procedure Authentication_ModExp(Var Authentication_, exp, modb, res :
TAuthentication_);
Procedure Authentication_ModBis(Const Authentication_ : TAuthentication_; Var
Authentication_Out : TAuthentication_; b : longint; head : int64);
Procedure Authentication_MulModBis(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Prod : TAuthentication_; b : longint; head : int64);
Procedure Authentication_MontgomeryMod(Const GInt, base, baseInv :
TAuthentication_; Var MGInt : TAuthentication_; b : longint; head : int64);
Procedure Authentication_MontgomeryModExp(Var Authentication_, exp, modb, res :
TAuthentication_);
Procedure Authentication_GCD(Const Authentication_1, Authentication_2 :
TAuthentication_; Var GCD : TAuthentication_);
Procedure Authentication_LCM(Const Authentication_1, Authentication_2 :
TAuthentication_; Var LCM : TAuthentication_);
Procedure Authentication_TrialDiv9999(Const Authentication_ : TAuthentication_;
Var ok : boolean);
Procedure Authentication_Random1(Var Seed, RandomAuthentication_ :
TAuthentication_);
Procedure Authentication_RabinMiller(Var Authentication_p : TAuthentication_;
nrtest : integer; Var ok : boolean);
Procedure Authentication_BezoutBachet(Var Authentication_1, Authentication_2, a,
b : TAuthentication_);
Procedure Authentication_ModInv(Const Authentication_1, base : TAuthentication_;
Var Inverse : TAuthentication_);
Procedure Authentication_Primetest(Var Authentication_p : TAuthentication_;
nrRMtests : integer; Var ok : boolean);
Procedure Authentication_LegendreSymbol(Var a, p : TAuthentication_; Var L :
integer);
Procedure Authentication_SquareRootModP(Square, Prime : TAuthentication_; Var
SquareRoot : TAuthentication_);



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  Authentication_

Procedure Base10StringToAuthentication_(Base10 : String; Var Authentication_ :
TAuthentication_);
Var
   i, size : longint;
   j : int64;
   S : String;
   sign : TSign;

   Procedure GIntDivByIntBis1(Var GInt : TAuthentication_; 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(Authentication_.Number, size + 1);
   Authentication_.Number[0] := size;
   For i := 1 To size - 1 Do
   Begin
      Authentication_.Number[i] := StrToInt(copy(Base10, length(Base10) - 8,
9));
      delete(Base10, length(Base10) - 8, 9);
   End;
   Authentication_.Number[size] := StrToInt(Base10);

   S := '';
   While (Authentication_.Number[0] <> 1) Or (Authentication_.Number[1] <> 0) Do
   Begin
      GIntDivByIntBis1(Authentication_, 2, j);
      S := inttostr(j) + S;
   End;
   S := '0' + S;
   Authentication_Destroy(Authentication_);
   Base2StringToAuthentication_(S, Authentication_);
   Authentication_.Sign := sign;
End;


 // перетворюємо  Authentication_ в рядок 10 біт

Procedure Authentication_ToBase10String(Const Authentication_ :
TAuthentication_; Var Base10 : String);
Var
   S : String;
   j : int64;
   temp : TAuthentication_;
Begin
   Authentication_Copy(Authentication_, temp);
   Base10 := '';
   While (temp.Number[0] > 1) Or (temp.Number[1] > 0) Do
   Begin
      Authentication_DivByIntBis(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 Authentication_.Sign = negative Then Base10 := '-' + Base10;
End;



// Видаляємо Authentication_ для звільнення пам'яті

Procedure Authentication_Destroy(Var Authentication_ : TAuthentication_);
Begin
   Authentication_.Number := Nil;
End;



Function Authentication_CompareAbs(Const Authentication_1, Authentication_2 :
TAuthentication_) : TCompare;
Var
   size1, size2, i : longint;
Begin
   Authentication_CompareAbs := Er;
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.Number[0];
   If size1 > size2 Then Authentication_CompareAbs := Lt Else
      If size1 < size2 Then Authentication_CompareAbs := St Else
      Begin
         i := size2;
         While (Authentication_1.Number[i] = Authentication_2.Number[i]) And (i
> 1) Do i := i - 1;
         If Authentication_1.Number[i] = Authentication_2.Number[i] Then
Authentication_CompareAbs := Eq Else
            If Authentication_1.Number[i] < Authentication_2.Number[i] Then
Authentication_CompareAbs := St Else
               If Authentication_1.Number[i] > Authentication_2.Number[i] Then
Authentication_CompareAbs := Lt;
      End;
End;


// Додаємо 2 Authentication_s, Authentication_1 + Authentication_2 = Sum

Procedure Authentication_Add(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Sum : TAuthentication_);
Var
   i, size1, size2, size : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.Number[0];
   If size1 < size2 Then Authentication_Add(Authentication_2, Authentication_1,
Sum) Else
   Begin
      If Authentication_1.Sign = Authentication_2.Sign Then
      Begin
         Sum.Sign := Authentication_1.Sign;
         SetLength(Sum.Number, size1 + 2);
         rest := 0;
         For i := 1 To size2 Do
         Begin
            Trest := Authentication_1.Number[i] + Authentication_2.Number[i] +
rest;
            Sum.Number[i] := Trest And 2147483647;
            rest := Trest Shr 31;
         End;
```

```
         For i := (size2 + 1) To size1 Do
         Begin
            Trest := Authentication_1.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 Authentication_CompareAbs(Authentication_2, Authentication_1) = Lt
Then Authentication_Add(Authentication_2, Authentication_1, Sum)
         Else
         Begin
            SetLength(Sum.Number, size1 + 1);
            rest := 0;
            For i := 1 To size2 Do
            Begin
               Trest := 2147483648 + Authentication_1.Number[i] -
Authentication_2.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 + Authentication_1.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 := Authentication_1.Sign;
         End;
      End;
   End;
End;


Procedure Authentication_ChangeSign(Var Authentication_ : TAuthentication_);
Begin
   If Authentication_.Sign = negative Then Authentication_.Sign := positive Else
Authentication_.Sign := negative;
End;


// Віднімаємо 2 Authentication_s, Authentication_1 - Authentication_2 = dif

Procedure Authentication_Sub(Var Authentication_1, Authentication_2, dif :
TAuthentication_);
Begin
   Authentication_ChangeSign(Authentication_2);
   Authentication_Add(Authentication_1, Authentication_2, dif);
   Authentication_ChangeSign(Authentication_2);
End;


// Перемножуємоа Authentication_ з цілим, Authentication_ * by = res, by <
1000000000
```

```
Procedure Authentication_MulByInt(Const Authentication_ : TAuthentication_; Var
res : TAuthentication_; by : int64);
Var
   i, size : longint;
   Trest, rest : int64;
Begin
   size := Authentication_.Number[0];
   SetLength(res.Number, size + 2);
   rest := 0;
   For i := 1 To size Do
   Begin
      Trest := Authentication_.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 := Authentication_.Sign;
End;


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

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


// ділимо Authentication_ на ціле, Authentication_ = res * by + modres

Procedure Authentication_DivByInt(Const Authentication_ : TAuthentication_; Var
res : TAuthentication_; by : int64; Var modres : int64);
Var
   i, size : longint;
   rest : int64;
Begin
   size := Authentication_.Number[0];
   SetLength(res.Number, size + 1);
   modres := 0;
   For i := size Downto 1 Do
   Begin
      modres := modres Shl 31;
      rest := modres Or Authentication_.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 := Authentication_.Sign;
End;


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

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


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

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



// повертаємо Authentication_ по модулю

Procedure Authentication_Abs(Var Authentication_ : TAuthentication_);
Begin
   Authentication_.Sign := positive;
End;

// Копіюємо Authentication_1 в Authentication_2
```

```
Procedure Authentication_Copy(Const Authentication_1 : TAuthentication_; Var
Authentication_2 : TAuthentication_);
Begin
   Authentication_2.Sign := Authentication_1.Sign;
   Authentication_2.Number := Nil;
   Authentication_2.Number := Copy(Authentication_1.Number, 0,
Authentication_1.Number[0] + 1);
End;

// Зрушуємо Authentication_ уліво на 2 біта Authentication_ = Authentication_ *
2

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

// Зрушуємо Authentication_ вправо на 2 біта, Authentication_ = Authentication_
div 2

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


// Authentication_ = Authentication_ / 2147483648

Procedure Authentication_ShiftRightBy31(Var Authentication_ : TAuthentication_);
Var
   size : longint;
Begin
   size := Authentication_.Number[0];
   If size > 1 Then
   Begin
      Authentication_.Number := Copy(Authentication_.Number, 1, Size);
      Authentication_.Number[0] := size - 1;
```

```
   End
   Else Authentication_.Number[1] := 0;
End;


// Authentication_1 = Authentication_1 + Authentication_2, Authentication_1 >
Authentication_2

Procedure Authentication_AddBis(Var Authentication_1 : TAuthentication_; Const
Authentication_2 : TAuthentication_);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := Authentication_1.Number[i] + Authentication_2.Number[i] + rest;
      rest := Trest Shr 31;
      Authentication_1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := Authentication_1.Number[i] + rest;
      rest := Trest Shr 31;
      Authentication_1.Number[i] := Trest And 2147483647;
   End;
   If rest <> 0 Then
   Begin
      SetLength(Authentication_1.Number, size1 + 2);
      Authentication_1.Number[0] := size1 + 1;
      Authentication_1.Number[size1 + 1] := rest;
   End;
End;


// Authentication_1 = Authentication_1 - Authentication_2, використовується
тільки якщо 0 < Authentication_2 < Authentication_1

Procedure Authentication_SubBis(Var Authentication_1 : TAuthentication_; Const
Authentication_2 : TAuthentication_);
Var
   i, size1, size2 : longint;
   rest : integer;
   Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.Number[0];
   rest := 0;
   For i := 1 To size2 Do
   Begin
      Trest := 2147483648 + Authentication_1.Number[i] -
Authentication_2.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      Authentication_1.Number[i] := Trest And 2147483647;
   End;
   For i := size2 + 1 To size1 Do
   Begin
      Trest := 2147483648 + Authentication_1.Number[i] + rest;
      If (Trest > 2147483647) Then rest := 0 Else rest := -1;
      Authentication_1.Number[i] := Trest And 2147483647;
   End;
   i := size1;
   While (Authentication_1.Number[i] = 0) And (i > 1) Do i := i - 1;
   If i < size1 Then
   Begin
```

```
      SetLength(Authentication_1.Number, i + 1);
      Authentication_1.Number[0] := i;
   End;
End;


// Перемножуємо 2 Authentication_s, Authentication_1 * Authentication_2 = Prod

Procedure Authentication_Mul(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Prod : TAuthentication_);
Var
   i, j, size, size1, size2 : longint;
   rest, Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.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] + Authentication_1.Number[j] *
Authentication_2.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 Authentication_1.Sign = Authentication_2.Sign Then Prod.Sign := Positive
Else prod.Sign := negative;
End;


// Підводимо в квадрат Authentication_, Authentication_I = Square

Procedure Authentication_Square(Const Authentication_ : TAuthentication_; Var
Square : TAuthentication_);
Var
   size, size1, i, j : longint;
   rest, Trest : int64;
Begin
   size1 := Authentication_.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] + Authentication_.Number[i] *
Authentication_.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 * Authentication_.Number[i] *
Authentication_.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 Authentication_ у бінарну рядок

Procedure Authentication_ToBase2String(Const Authentication_ : TAuthentication_;
Var S : String);
Var
   i : longint;
   j : integer;
Begin
   S := '';
   For i := 1 To Authentication_.Number[0] Do
   Begin
      For j := 0 To 30 Do S := inttostr(1 And (Authentication_.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 Base2StringToAuthentication_(S : String; Var Authentication_ :
TAuthentication_);
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(Authentication_.Number, size + 1);
   Authentication_.Number[0] := size;
   j := 1;
   Authentication_.Number[j] := 0;
   i := 0;
   While length(S) > 0 Do
   Begin
      If S[length(S)] = '1' Then
         Authentication_.Number[j] := Authentication_.Number[j] Or (1 Shl i);
      i := i + 1;
      If i = 31 Then
      Begin
         i := 0;
         j := j + 1;
         If j <= size Then Authentication_.Number[j] := 0;
      End;
      delete(S, length(S), 1);
   End;
   Authentication_.Sign := positive;
End;


// перетврюємо Authentication_ у рядок 256 біт

Procedure Authentication_ToBase256String(Const Authentication_ :
TAuthentication_; Var str256 : String);
Var
   temp1 : String;
```

```
   i, len8 : longint;
   g : char;
Begin
   Authentication_ToBase2String(Authentication_, 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 Base256StringToAuthentication_(str256 : String; Var Authentication_ :
TAuthentication_);
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);
   Base2StringToAuthentication_(temp1, Authentication_);
End;

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

Procedure PGPMPIToAuthentication_(PGPMPI : String; Var Authentication_ :
TAuthentication_);
Var
   temp : String;
Begin
   temp := PGPMPI;
   delete(temp, 1, 2);
   Base256StringToAuthentication_(temp, Authentication_);
End;


Procedure Authentication_ToPGPMPI(Authentication_ : TAuthentication_; Var PGPMPI
: String);
Var
   len, i : word;
   c : char;
   b : byte;
Begin
   Authentication_ToBase256String(Authentication_, 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 Authentication_, Authentication_^exp = res

Procedure Authentication_Exp(Const Authentication_, exp : TAuthentication_; Var
res : TAuthentication_);
Var
   temp2, temp3 : TAuthentication_;
```

```
   S : String;
   i : longint;
Begin
   Authentication_ToBase2String(exp, S);
   If S[length(S)] = '0' Then Base10StringToAuthentication_('1', res) Else
Authentication_Copy(Authentication_, res);
   Authentication_Copy(Authentication_, temp2);
   If length(S) > 1 Then
      For i := (length(S) - 1) Downto 1 Do
      Begin
         Authentication_Square(temp2, temp3);
         Authentication_Copy(temp3, temp2);
         If S[i] = '1' Then
         Begin
            Authentication_Mul(res, temp2, temp3);
            Authentication_Copy(temp3, res);
         End;
      End;
End;


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

Procedure Authentication_Fac(Const Authentication_ : TAuthentication_; Var res :
TAuthentication_);
Var
   one, temp, temp1 : TAuthentication_;
Begin
   Authentication_Copy(Authentication_, temp);
   Base10StringToAuthentication_('1', res);
   Base10StringToAuthentication_('1', one);

   While Not (Authentication_CompareAbs(temp, one) = Eq) Do
   Begin
      Authentication_Mul(temp, res, temp1);
      Authentication_Copy(temp1, res);
      Authentication_SubBis(temp, one);
   End;

   Authentication_Destroy(one);
   Authentication_Destroy(temp);
End;


// Authentication_ = Authentication_ * 2147483648

Procedure Authentication_ShiftLeftBy31(Var Authentication_ : TAuthentication_);
Var
   f1, f2 : int64;
   i, size : longint;
Begin
   size := Authentication_.Number[0];
   SetLength(Authentication_.Number, size + 2);
   f1 := 0;
   For i := 1 To (size + 1) Do
   Begin
      f2 := Authentication_.Number[i];
      Authentication_.Number[i] := f1;
      f1 := f2;
   End;
   Authentication_.Number[0] := size + 1;
End;


// Ділимо 2 Authentication_s, Authentication_1 = Authentication_2 *
QAuthentication_ + MAuthentication_, MAuthentication_ позитивне
```

```
Procedure Authentication_DivMod(Var Authentication_1, Authentication_2,
QAuthentication_, MAuthentication_ : TAuthentication_);
Var
   one, zero, temp1, temp2 : TAuthentication_;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := Authentication_1.Sign;
   s2 := Authentication_2.Sign;
   Authentication_Abs(Authentication_1);
   Authentication_Abs(Authentication_2);
   Authentication_Copy(Authentication_1, MAuthentication_);
   Authentication_Copy(Authentication_2, temp1);

   If Authentication_CompareAbs(Authentication_1, Authentication_2) <> St Then
   Begin
      s := Authentication_1.Number[0] - Authentication_2.Number[0];
      setlength(QAuthentication_.Number, s + 2);
      QAuthentication_.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         Authentication_ShiftLeftBy31(temp1);
         QAuthentication_.Number[t] := 0;
      End;
      j := s + 1;
      QAuthentication_.Number[j] := 0;
      While Authentication_CompareAbs(MAuthentication_, Authentication_2) <> St
Do
      Begin
         While Authentication_CompareAbs(MAuthentication_, temp1) <> St Do
         Begin
            If MAuthentication_.Number[0] > temp1.Number[0] Then
               i := (2147483648 *
MAuthentication_.Number[MAuthentication_.Number[0]] +
MAuthentication_.Number[MAuthentication_.Number[0] - 1]) Div
(temp1.Number[temp1.Number[0]] + 1)
            Else i := MAuthentication_.Number[MAuthentication_.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               Authentication_Copy(temp1, temp2);
               Authentication_MulByIntBis(temp2, i);
               Authentication_SubBis(MAuthentication_, temp2);
               QAuthentication_.Number[j] := QAuthentication_.Number[j] + i;
               If Authentication_CompareAbs(MAuthentication_, temp2) <> St Then
               Begin
                  QAuthentication_.Number[j] := QAuthentication_.Number[j] + i;
                  Authentication_SubBis(MAuthentication_, temp2);
               End;
            End Else
            Begin
               QAuthentication_.Number[j] := QAuthentication_.Number[j] + 1;
               Authentication_SubBis(MAuthentication_, temp1);
            End;
         End;
         If MAuthentication_.Number[0] <= temp1.Number[0] Then
            If Authentication_CompareAbs(temp1, Authentication_2) <> Eq Then
            Begin
               Authentication_ShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
   Else Base10StringToAuthentication_('0', QAuthentication_);
   s := QAuthentication_.Number[0];
   While (s > 1) And (QAuthentication_.Number[s] = 0) Do s := s - 1;
   If s < QAuthentication_.Number[0] Then
   Begin
      setlength(QAuthentication_.Number, s + 1);
```

```
      QAuthentication_.Number[0] := s;
   End;
   QAuthentication_.Sign := positive;

   Authentication_Destroy(temp1);
   Base10StringToAuthentication_('0', zero);
   Base10StringToAuthentication_('1', one);
   If s1 = negative Then
   Begin
      If Authentication_CompareAbs(MAuthentication_, zero) <> Eq Then
      Begin
         Authentication_add(QAuthentication_, one, temp1);
         Authentication_Copy(temp1, QAuthentication_);
         Authentication_Destroy(temp1);
         Authentication_sub(Authentication_2, MAuthentication_, temp1);
         Authentication_Copy(temp1, MAuthentication_);
      End;
      If s2 = positive Then QAuthentication_.Sign := negative;
   End
   Else QAuthentication_.Sign := s2;
   Authentication_Destroy(one);
   Authentication_Destroy(zero);

   Authentication_1.Sign := s1;
   Authentication_2.Sign := s2;
End;


// Розраховуємо MAuthentication_

Procedure Authentication_Div(Var Authentication_1, Authentication_2,
QAuthentication_ : TAuthentication_);
Var
   one, zero, temp1, temp2, MAuthentication_ : TAuthentication_;
   s1, s2 : TSign;
   i, j, s, t : longint;
Begin
   s1 := Authentication_1.Sign;
   s2 := Authentication_2.Sign;
   Authentication_Abs(Authentication_1);
   Authentication_Abs(Authentication_2);
   Authentication_Copy(Authentication_1, MAuthentication_);
   Authentication_Copy(Authentication_2, temp1);

   If Authentication_CompareAbs(Authentication_1, Authentication_2) <> St Then
   Begin
      s := Authentication_1.Number[0] - Authentication_2.Number[0];
      setlength(QAuthentication_.Number, s + 2);
      QAuthentication_.Number[0] := s + 1;
      For t := 1 To s Do
      Begin
         Authentication_ShiftLeftBy31(temp1);
         QAuthentication_.Number[t] := 0;
      End;
      j := s + 1;
      QAuthentication_.Number[j] := 0;
      While Authentication_CompareAbs(MAuthentication_, Authentication_2) <> St
Do
      Begin
         While Authentication_CompareAbs(MAuthentication_, temp1) <> St Do
         Begin
            If MAuthentication_.Number[0] > temp1.Number[0] Then
               i := (2147483648 *
MAuthentication_.Number[MAuthentication_.Number[0]] +
MAuthentication_.Number[MAuthentication_.Number[0] - 1]) Div
(temp1.Number[temp1.Number[0]] + 1)
            Else i := MAuthentication_.Number[MAuthentication_.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
```

```
            Begin
               Authentication_Copy(temp1, temp2);
               Authentication_MulByIntBis(temp2, i);
               Authentication_SubBis(MAuthentication_, temp2);
               QAuthentication_.Number[j] := QAuthentication_.Number[j] + i;
               If Authentication_CompareAbs(MAuthentication_, temp2) <> St Then
               Begin
                  QAuthentication_.Number[j] := QAuthentication_.Number[j] + i;
                  Authentication_SubBis(MAuthentication_, temp2);
               End;
            End Else
            Begin
               QAuthentication_.Number[j] := QAuthentication_.Number[j] + 1;
               Authentication_SubBis(MAuthentication_, temp1);
            End;
         End;
         If MAuthentication_.Number[0] <= temp1.Number[0] Then
            If Authentication_CompareAbs(temp1, Authentication_2) <> Eq Then
            Begin
               Authentication_ShiftRightBy31(temp1);
               j := j - 1;
            End;
      End;
   End
   Else Base10StringToAuthentication_('0', QAuthentication_);
   s := QAuthentication_.Number[0];
   While (s > 1) And (QAuthentication_.Number[s] = 0) Do s := s - 1;
   If s < QAuthentication_.Number[0] Then
   Begin
      setlength(QAuthentication_.Number, s + 1);
      QAuthentication_.Number[0] := s;
   End;
   QAuthentication_.Sign := positive;

   Authentication_Destroy(temp1);
   Base10StringToAuthentication_('0', zero);
   Base10StringToAuthentication_('1', one);
   If s1 = negative Then
   Begin
      If Authentication_CompareAbs(MAuthentication_, zero) <> Eq Then
      Begin
         Authentication_add(QAuthentication_, one, temp1);
         Authentication_Copy(temp1, QAuthentication_);
         Authentication_Destroy(temp1);
         Authentication_sub(Authentication_2, MAuthentication_, temp1);
         Authentication_Copy(temp1, MAuthentication_);
      End;
      If s2 = positive Then QAuthentication_.Sign := negative;
   End
   Else QAuthentication_.Sign := s2;
   Authentication_Destroy(one);
   Authentication_Destroy(zero);
   Authentication_Destroy(MAuthentication_);

   Authentication_1.Sign := s1;
   Authentication_2.Sign := s2;
End;


// MAuthentication_ = Authentication_1 mod Authentication_2

Procedure Authentication_Mod(Var Authentication_1, Authentication_2,
MAuthentication_ : TAuthentication_);
Var
   one, zero, temp1, temp2 : TAuthentication_;
   s1, s2 : TSign;
   i : int64;
   s, t : longint;
Begin
```

```
   s1 := Authentication_1.Sign;
   s2 := Authentication_2.Sign;
   Authentication_Abs(Authentication_1);
   Authentication_Abs(Authentication_2);
   Authentication_Copy(Authentication_1, MAuthentication_);
   Authentication_Copy(Authentication_2, temp1);

   If Authentication_CompareAbs(Authentication_1, Authentication_2) <> St Then
   Begin
      s := Authentication_1.Number[0] - Authentication_2.Number[0];
      For t := 1 To s Do Authentication_ShiftLeftBy31(temp1);
      While Authentication_CompareAbs(MAuthentication_, Authentication_2) <> St
Do
      Begin
         While Authentication_CompareAbs(MAuthentication_, temp1) <> St Do
         Begin
            If MAuthentication_.Number[0] > temp1.Number[0] Then
               i := (2147483648 *
MAuthentication_.Number[MAuthentication_.Number[0]] +
MAuthentication_.Number[MAuthentication_.Number[0] - 1]) Div
(temp1.Number[temp1.Number[0]] + 1)
            Else i := MAuthentication_.Number[MAuthentication_.Number[0]] Div
(temp1.Number[temp1.Number[0]] + 1);
            If (i <> 0) Then
            Begin
               Authentication_Copy(temp1, temp2);
               Authentication_MulByIntBis(temp2, i);
               Authentication_SubBis(MAuthentication_, temp2);
               If Authentication_CompareAbs(MAuthentication_, temp2) <> St Then
Authentication_SubBis(MAuthentication_, temp2);
            End Else Authentication_SubBis(MAuthentication_, temp1);
//         If Authentication_CompareAbs(MAuthentication_, temp1) <> St Then
Authentication_SubBis(MAuthentication_,temp1);
         End;
         If MAuthentication_.Number[0] <= temp1.Number[0] Then
            If Authentication_CompareAbs(temp1, Authentication_2) <> Eq Then
Authentication_ShiftRightBy31(temp1);
      End;
   End;

   Authentication_Destroy(temp1);
   Base10StringToAuthentication_('0', zero);
   Base10StringToAuthentication_('1', one);
   If s1 = negative Then
   Begin
      If Authentication_CompareAbs(MAuthentication_, zero) <> Eq Then
      Begin
         Authentication_Sub(Authentication_2, MAuthentication_, temp1);
         Authentication_Copy(temp1, MAuthentication_);
      End;
   End;
   Authentication_Destroy(one);
   Authentication_Destroy(zero);

   Authentication_1.Sign := s1;
   Authentication_2.Sign := s2;
End;


// підводимо у квадрат Authentication_ за модулем  Modb, Authentication_^2 mod
Modb = Authentication_SM

Procedure Authentication_SquareMod(Var Authentication_, Modb, Authentication_SM
: TAuthentication_);
Var
   temp : TAuthentication_;
Begin
   Authentication_Square(Authentication_, temp);
   Authentication_Mod(temp, Modb, Authentication_SM);
```

```
   Authentication_Destroy(temp);
End;


// Додаємо 2 Authentication_s за модулем, (Authentication_1 + Authentication_2)
mod base = Authentication_res

Procedure Authentication_AddMod(Var Authentication_1, Authentication_2, base,
Authentication_res : TAuthentication_);
Var
   temp : TAuthentication_;
Begin
   Authentication_add(Authentication_1, Authentication_2, temp);
   Authentication_Mod(temp, base, Authentication_res);
   Authentication_Destroy(temp);
End;


// Перемножуємо 2 Authentication_s за модулем, (Authentication_1 *
Authentication_2) mod base = Authentication_res

Procedure Authentication_MulMod(Var Authentication_1, Authentication_2, base,
Authentication_res : TAuthentication_);
Var
   temp : TAuthentication_;
Begin
   Authentication_Mul(Authentication_1, Authentication_2, temp);
   Authentication_Mod(temp, base, Authentication_res);
   Authentication_Destroy(temp);
End;


// Підводимо у ступінь 2 Authentication_s за модулем, (Authentication_1 ^
Authentication_2) mod modb = res

Procedure Authentication_ModExp(Var Authentication_, exp, modb, res :
TAuthentication_);
Var
   temp2, temp3 : TAuthentication_;
   i : longint;
   S : String;
Begin
   If (Modb.Number[1] Mod 2) = 1 Then
   Begin
      Authentication_MontgomeryModExp(Authentication_, exp, modb, res);
      exit;
   End;
   Authentication_ToBase2String(exp, S);
   Base10StringToAuthentication_('1', res);
   Authentication_copy(Authentication_, temp2);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         Authentication_mulMod(res, temp2, modb, temp3);
         Authentication_Copy(temp3, res);
      End;
      Authentication_SquareMod(temp2, Modb, temp3);
      Authentication_Copy(temp3, temp2);
   End;
   Authentication_Destroy(temp2);
End;


// Процедура підводення у ступень за Монтгомері

Procedure Authentication_ModBis(Const Authentication_ : TAuthentication_; Var
Authentication_Out : TAuthentication_; b : longint; head : int64);
```

```
Var
   i : longint;
Begin
   If b <= Authentication_.Number[0] Then
   Begin
      Authentication_Out.Number := Copy(Authentication_.Number, 0, b + 1);
      Authentication_Out.Number[b] := Authentication_Out.Number[b] And head;
      i := b;
      While (Authentication_Out.Number[i] = 0) And (i > 1) Do i := i - 1;
      If i < b Then SetLength(Authentication_Out.Number, i + 1);
      Authentication_Out.Number[0] := i;
      Authentication_Out.Sign := positive;
   End Else Authentication_Copy(Authentication_, Authentication_Out);
End;


Procedure Authentication_MulModBis(Const Authentication_1, Authentication_2 :
TAuthentication_; Var Prod : TAuthentication_; b : longint; head : int64);
Var
   i, j, size, size1, size2, t : longint;
   rest, Trest : int64;
Begin
   size1 := Authentication_1.Number[0];
   size2 := Authentication_2.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] + Authentication_1.Number[j] *
Authentication_2.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 Authentication_1.Sign = Authentication_2.Sign Then Prod.Sign := Positive
Else prod.Sign := negative;
End;


Procedure Authentication_MontgomeryMod(Const GInt, base, baseInv :
TAuthentication_; Var MGInt : TAuthentication_; b : longint; head : int64);
Var
   m, temp, temp1 : TAuthentication_;
   r : int64;
Begin
   Authentication_ModBis(GInt, temp, b, head);
   Authentication_MulModBis(temp, baseInv, m, b, head);
   Authentication_Mul(m, base, temp1);
   Authentication_Destroy(temp);
   Authentication_Add(temp1, GInt, temp);
   Authentication_Destroy(temp1);
   MGInt.Number := copy(temp.Number, b - 1, temp.Number[0] - b + 2);
   MGInt.Sign := positive;
```

```
   MGInt.Number[0] := temp.Number[0] - b + 1;
   Authentication_Destroy(temp);
   If (head Shr 30) = 0 Then Authentication_DivByIntBis(MGInt, head + 1, r)
   Else Authentication_ShiftRightBy31(MGInt);
   If Authentication_CompareAbs(MGInt, base) <> St Then
Authentication_SubBis(MGInt, base);
   Authentication_Destroy(temp);
   Authentication_Destroy(m);
End;


Procedure Authentication_MontgomeryModExp(Var Authentication_, exp, modb, res :
TAuthentication_);
Var
   temp2, temp3, baseInv, r : TAuthentication_;
   i, j, t, b : longint;
   S : String;
   head : int64;
Begin
   Authentication_ToBase2String(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;

   Authentication_ModInv(modb, r, temp2);
   If temp2.Sign = negative Then Authentication_Copy(temp2, BaseInv)
   Else
   Begin
      Authentication_Copy(r, BaseInv);
      Authentication_SubBis(BaseInv, temp2);
   End;
 //  Authentication_BezoutBachet(r, modb, temp2, BaseInv);
   Authentication_Abs(BaseInv);
   Authentication_Destroy(temp2);
   Authentication_Mod(r, modb, res);
   Authentication_MulMod(Authentication_, res, modb, temp2);
   Authentication_Destroy(r);

   For i := length(S) Downto 1 Do
   Begin
      If S[i] = '1' Then
      Begin
         Authentication_mul(res, temp2, temp3);
         Authentication_Destroy(res);
         Authentication_MontgomeryMod(temp3, modb, baseinv, res, b, head);
         Authentication_Destroy(temp3);
```

```
      End;
      Authentication_Square(temp2, temp3);
      Authentication_Destroy(temp2);
      Authentication_MontgomeryMod(temp3, modb, baseinv, temp2, b, head);
      Authentication_Destroy(temp3);
   End;
   Authentication_Destroy(temp2);
   Authentication_MontgomeryMod(res, modb, baseinv, temp3, b, head);
   Authentication_Copy(temp3, res);
   Authentication_Destroy(temp3);
   Authentication_Destroy(baseinv);
End;


// розраховуємо найбільший загальний дільник 2 Authentication_s

Procedure Authentication_GCD(Const Authentication_1, Authentication_2 :
TAuthentication_; Var GCD : TAuthentication_);
Var
   k : TCompare;
   zero, temp1, temp2, temp3 : TAuthentication_;
Begin
   k := Authentication_CompareAbs(Authentication_1, Authentication_2);
   If (k = Eq) Then Authentication_Copy(Authentication_1, GCD) Else
      If (k = St) Then Authentication_GCD(Authentication_2, Authentication_1,
GCD) Else
      Begin
         Base10StringToAuthentication_('0', zero);
         Authentication_Copy(Authentication_1, temp1);
         Authentication_Copy(Authentication_2, temp2);
         While (temp2.Number[0] <> 1) Or (temp2.Number[1] <> 0) Do
         Begin
            Authentication_Mod(temp1, temp2, temp3);
            Authentication_Copy(temp2, temp1);
            Authentication_Copy(temp3, temp2);
            Authentication_Destroy(temp3);
         End;
         Authentication_Copy(temp1, GCD);
         Authentication_Destroy(temp2);
         Authentication_Destroy(zero);
      End;
End;


// розраховуємо найменше загальне кратне 2 Authentication_s

Procedure Authentication_LCM(Const Authentication_1, Authentication_2 :
TAuthentication_; Var LCM : TAuthentication_);
Var
   temp1, temp2 : TAuthentication_;
Begin
   Authentication_GCD(Authentication_1, Authentication_2, temp1);
   Authentication_mul(Authentication_1, Authentication_2, temp2);
   Authentication_div(temp2, temp1, LCM);
   Authentication_Destroy(temp1);
   Authentication_Destroy(temp2);
End;


// Знаходження взаємо простого Authentication_ до 9999 та зупинка коли знайдено
таке число, повертає ok=false

Procedure Authentication_TrialDiv9999(Const Authentication_ : TAuthentication_;
Var ok : boolean);
Var
   j : int64;
   i : integer;
Begin
   If ((Authentication_.Number[1] Mod 2) = 0) Then ok := false
```

```
   Else
   Begin
      i := 0;
      ok := true;
      While ok And (i < 1228) Do
      Begin
         i := i + 1;
         Authentication_modbyint(Authentication_, primes[i], j);
         If j = 0 Then ok := false;
      End;
   End;
End;
// Генератор випадкових чисел

Procedure Authentication_Random1(Var Seed, RandomAuthentication_ :
TAuthentication_);
Var
   temp, base : TAuthentication_;
Begin
   Base10StringToAuthentication_('281474976710656', base);
   Base10StringToAuthentication_('44485709377909', temp);
   Authentication_MulMod(seed, temp, base, RandomAuthentication_);
   Authentication_Destroy(temp);
   Authentication_Destroy(base);
End;


// Тест на простоту числа Authentication_p методом Рабіна-Мілера, повертає
ok=true якщо Authentication_p пройшло тест

Procedure Authentication_RabinMiller(Var Authentication_p : TAuthentication_;
nrtest : integer; Var ok : boolean);
Var
   j, b, i : int64;
   m, z, temp1, temp2, temp3, zero, one, two, pmin1 : TAuthentication_;
   ok1, ok2 : boolean;
Begin
   randomize;
   j := 0;
   Base10StringToAuthentication_('0', zero);
   Base10StringToAuthentication_('1', one);
   Base10StringToAuthentication_('2', two);
   Authentication_sub(Authentication_p, one, temp1);
   Authentication_sub(Authentication_p, one, pmin1);

   b := 0;
   While (temp1.Number[1] Mod 2) = 0 Do
   Begin
      b := b + 1;
      Authentication_ShiftRight(temp1);
   End;
   m := temp1;

   i := 0;
   ok := true;
   Randomize;
   While (i < nrtest) And ok Do
   Begin
      i := i + 1;
      Base10StringToAuthentication_(inttostr(Primes[Random(1227) + 1]), temp2);
      Authentication_MontGomeryModExp(temp2, m, Authentication_p, z);
      Authentication_Destroy(temp2);
      ok1 := (Authentication_CompareAbs(z, one) = Eq);
      ok2 := (Authentication_CompareAbs(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
                  Authentication_Squaremod(z, Authentication_p, temp3);
                  Authentication_Copy(temp3, z);
                  ok1 := (Authentication_CompareAbs(z, one) = Eq);
                  ok2 := (Authentication_CompareAbs(z, pmin1) = Eq);
                  If ok2 Then j := b;
               End
               Else If (Not ok2) And (j >= b) Then ok := false;
            End;
         End;

      End
   End;

   Authentication_Destroy(zero);
   Authentication_Destroy(one);
   Authentication_Destroy(two);
   Authentication_Destroy(m);
   Authentication_Destroy(z);
   Authentication_Destroy(pmin1);
End;


// Розраховуємо коефіцієнти з теореми Безу, Authentication_1 * a +
Authentication_2 * b = GCD(Authentication_1, Authentication_2)

Procedure Authentication_BezoutBachet(Var Authentication_1, Authentication_2, a,
b : TAuthentication_);
Var
   zero, r1, r2, r3, ta, gcd, temp, temp1, temp2 : TAuthentication_;
Begin
   If Authentication_CompareAbs(Authentication_1, Authentication_2) <> St Then
   Begin
      Authentication_copy(Authentication_1, r1);
      Authentication_copy(Authentication_2, r2);
      Base10StringToAuthentication_('0', zero);
      Base10StringToAuthentication_('1', a);
      Base10StringToAuthentication_('0', ta);

      Repeat
         Authentication_divmod(r1, r2, temp, r3);
         Authentication_Destroy(r1);
         r1 := r2;
         r2 := r3;

         Authentication_mul(ta, temp, temp1);
         Authentication_sub(a, temp1, temp2);
         Authentication_Copy(ta, a);
         Authentication_Copy(temp2, ta);
         Authentication_Destroy(temp1);

         Authentication_Destroy(temp);
      Until Authentication_CompareAbs(r3, zero) = Eq;

      Authentication_GCD(Authentication_1, Authentication_2, gcd);
      Authentication_mul(a, Authentication_1, temp1);
      Authentication_sub(gcd, temp1, temp2);
      Authentication_Destroy(temp1);
      Authentication_div(temp2, Authentication_2, b);
      Authentication_Destroy(temp2);

      Authentication_Destroy(ta);
      Authentication_Destroy(r1);
      Authentication_Destroy(r2);
```

```
      Authentication_Destroy(gcd);
   End
   Else Authentication_BezoutBachet(Authentication_2, Authentication_1, b, a);
End;


// Знахродимо мультпликативне зворотнє Authentication_ у кінцевому кільці
позитивного порядку

Procedure Authentication_ModInv(Const Authentication_1, base : TAuthentication_;
Var Inverse : TAuthentication_);
Var
   zero, one, r1, r2, r3, tb, gcd, temp, temp1, temp2 : TAuthentication_;
Begin
   Base10StringToAuthentication_('1', one);
   Authentication_GCD(Authentication_1, base, gcd);
   If Authentication_CompareAbs(one, gcd) = Eq Then
   Begin
      Authentication_copy(base, r1);
      Authentication_copy(Authentication_1, r2);
      Base10StringToAuthentication_('0', zero);
      Base10StringToAuthentication_('0', inverse);
      Base10StringToAuthentication_('1', tb);

      Repeat
         Authentication_Destroy(r3);
         Authentication_divmod(r1, r2, temp, r3);
         Authentication_Copy(r2, r1);
         Authentication_Copy(r3, r2);

         Authentication_mul(tb, temp, temp1);
         Authentication_sub(inverse, temp1, temp2);
         Authentication_Destroy(inverse);
         Authentication_Destroy(temp1);
         Authentication_Copy(tb, inverse);
         Authentication_Copy(temp2, tb);

         Authentication_Destroy(temp);
      Until Authentication_CompareAbs(r3, zero) = Eq;

      If inverse.Sign = negative Then
      Begin
         Authentication_add(base, inverse, temp);
         Authentication_Copy(temp, inverse);
      End;

      Authentication_Destroy(tb);
      Authentication_Destroy(r1);
      Authentication_Destroy(r2);
   End;
   Authentication_Destroy(gcd);
   Authentication_Destroy(one);
End;


// Простй комбінований тест на простоту Authentication_p

Procedure Authentication_Primetest(Var Authentication_p : TAuthentication_;
nrRMtests : integer; Var ok : boolean);
Begin
   Authentication_Trialdiv9999(Authentication_p, ok);
   If ok Then Authentication_RabinMiller(Authentication_p, nrRMtests, ok);
End;


//Розрахунок символу Лагранжа

Procedure Authentication_LegendreSymbol(Var a, p : TAuthentication_; Var L :
integer);
```

```
Var
   temp1, temp2, temp3, temp4, temp5, zero, one : TAuthentication_;
   i : int64;
   ok1, ok2 : boolean;
Begin
   Base10StringToAuthentication_('0', zero);
   Base10StringToAuthentication_('1', one);
   Authentication_Mod(a, p, temp1);
   If Authentication_CompareAbs(zero, temp1) = Eq Then
   Begin
      Authentication_Destroy(temp1);
      L := 0;
   End
   Else
   Begin
      Authentication_Destroy(temp1);
      Authentication_Copy(p, temp1);
      Authentication_Copy(a, temp2);
      L := 1;
      While Authentication_CompareAbs(temp2, one) <> Eq Do
      Begin
         If (temp2.Number[1] Mod 2) = 0 Then
         Begin
            Authentication_Square(temp1, temp3);
            Authentication_Sub(temp3, one, temp4);
            Authentication_Destroy(temp3);
            Authentication_DivByInt(temp4, temp3, 8, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok1 := false Else ok1 := true;
            Authentication_Destroy(temp3);
            Authentication_Destroy(temp4);
            If ok1 = true Then L := L * (-1);
            Authentication_DivByIntBis(temp2, 2, i);
         End
         Else
         Begin
            Authentication_Sub(temp1, one, temp3);
            Authentication_Sub(temp2, one, temp4);
            Authentication_Mul(temp3, temp4, temp5);
            Authentication_Destroy(temp3);
            Authentication_Destroy(temp4);
            Authentication_DivByInt(temp5, temp3, 4, i);
            If (temp3.Number[1] Mod 2) = 0 Then ok2 := false Else ok2 := true;
            Authentication_Destroy(temp5);
            Authentication_Destroy(temp3);
            If ok2 = true Then L := L * (-1);
            Authentication_Mod(temp1, temp2, temp3);
            Authentication_Copy(temp2, temp1);
            Authentication_Copy(temp3, temp2);
         End;
      End;
      Authentication_Destroy(temp1);
      Authentication_Destroy(temp2);
   End;
   Authentication_Destroy(zero);
   Authentication_Destroy(one);
End;


// Розрахунок квадратичного корня за модулем простого числа
// SquareRoot^2 mod Prime = Square

Procedure Authentication_SquareRootModP(Square, Prime : TAuthentication_; Var
SquareRoot : TAuthentication_);
Var
   one, n, b, s, r, temp, temp1, temp2, temp3 : TAuthentication_;
   a, L, i, j : longint;
Begin
   Base2StringToAuthentication_('1', one);
   Base2StringToAuthentication_('10', n);
```

```
    a := 0;
    Authentication_LegendreSymbol(n, Prime, L);
    While L <> -1 Do
    Begin
       Authentication_AddBis(n, one);
       Authentication_LegendreSymbol(n, Prime, L);
    End;
    Authentication_Copy(Prime, s);
    s.Number[1] := s.Number[1] - 1;
    While (s.Number[1] Mod 2) = 0 Do
    Begin
       Authentication_ShiftRight(s);
       a := a + 1;
    End;
    Authentication_MontgomeryModExp(n, s, Prime, b);
    Authentication_Add(s, one, temp);
    Authentication_ShiftRight(temp);
    Authentication_MontgomeryModExp(Square, temp, Prime, r);
    Authentication_Destroy(temp);
    Authentication_ModInv(Square, Prime, temp1);

    For i := 0 To (a - 2) Do
    Begin
       Authentication_SquareMod(r, Prime, temp2);
       Authentication_MulMod(temp1, temp2, Prime, temp);
       Authentication_Destroy(temp2);
       For j := 1 To (a - i - 2) Do
       Begin
          Authentication_SquareMod(temp, Prime, temp2);
          Authentication_Destroy(temp);
          Authentication_Copy(temp2, temp);
          Authentication_Destroy(temp2);
       End;
       If Authentication_CompareAbs(temp, one) <> Eq Then
       Begin
          Authentication_MulMod(r, b, Prime, temp3);
          Authentication_Destroy(r);
          Authentication_Copy(temp3, r);
          Authentication_Destroy(temp3);
       End;
       Authentication_Destroy(temp);
       Authentication_Destroy(temp2);
       If i = (a - 2) Then break;
       Authentication_SquareMod(b, Prime, temp3);
       Authentication_Destroy(b);
       Authentication_Copy(temp3, b);
       Authentication_Destroy(temp3);
    End;

    Authentication_Copy(r, SquareRoot);
    Authentication_Destroy(r);
    Authentication_Destroy(s);
    Authentication_Destroy(b);
    Authentication_Destroy(temp1);
    Authentication_Destroy(one);
    Authentication_Destroy(n);
End;


End.
```

Файл Authentication_PrimeGeneration.pas – генерація випадкових простих чисел для алгоритму RSA

```
Unit Authentication_PrimeGeneration;

Interface

Uses Windows, SysUtils, Controls, Authentication_;

Procedure PrimeSearch(Var GInt : TAuthentication_);

Implementation


{$H+}


// Відбувається пошаговий пошук простого числа починаючи з GInt,
// якщо воно знайдено то зберігається у GInt

Procedure PrimeSearch(Var GInt : TAuthentication_);
Var
   temp, two : TAuthentication_;
   ok : Boolean;
Begin
   If (GInt.Number[1] Mod 2) = 0 Then GInt.Number[1] := GInt.Number[1] + 1;
   Base10StringToAuthentication_('2', two);
   ok := false;
   While Not ok Do
   Begin
      Authentication_Add(GInt, two, temp);
      Authentication_Copy(temp, GInt);
      Authentication_PrimeTest(GInt, 4, ok);
   End;
   Authentication_Destroy(two);
End;

End.
```

Файл FleshKey.pas – робота з флеш-ключами

```
unit FleshKey;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Authentication_RSA, Authentication_RSA, Authentication_,
Authentication_PrimeGeneration
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 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('БАКАЛАВРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи кібербезпеки для автентифікації
користувача хмарного сервісу');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Смірнова Т.В.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Усатов Гліб Сергійович');
Memo1.Lines.Add('                  гр. КБ-19');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;
end.
```