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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Босенко Д.С.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 2024 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 2024 року

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

Босенку Денису Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700*

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 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Босенко Д.С.
(прізвище та ініціали)

АНОТАЦІЯ

Босенко Д.С. Програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Метою розробки є програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Результат роботи – програмна реалізація системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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 з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: кібербезпека, документообіг, AES Intel CORE I9-13900K S1700

ABSTRACT

Bosenko D.S. Secure document management cyber security system software using Intel CORE I9-13900K S1700 processor AES instructions. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of protected document circulation using the AES instructions of the Intel CORE I9-13900K S1700 processor.

The purpose of the development is the software of the cyber security system of the protected document flow using the AES instructions of the Intel CORE I9-13900K S1700 processor.

The result of the work is the software implementation of the cyber security system of protected document circulation using the AES instructions of the Intel CORE I9-13900K S1700 processor.

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 a PC with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: cyber security, document management, AES Intel CORE I9-13900K S1700

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСЗ	–	комплексна система захисту
ПЕОМ	–	персональна електронно-обчислювальна машина
СУБД	–	система управління базами даних
PGP	–	Pretty Good Privacy

ВСТУП

Актуальність теми. Сучасні організаційні системи частіше застосовують електронний документообіг, що став обов'язковим елементом інфраструктури інформаційних технологій. Завдяки електронному документообігу підвищується ефективність діяльність промислових підприємств і комерційних компаній, а в державних установах простіше вирішуються завдання взаємодії населення, міжвідомчої взаємодії й внутрішнього керування.

Але досить часто, інформація яка циркулює у системі електронного документообігу потребує захисту. Тому у даному бакалаврському проекті пропонується захищати цю інформацію з використанням команд Intel CORE I9-13900K S1700.

Intel CORE I9-13900K S1700 – це новий набір команд шифрування, що поліпшує алгоритм Advanced Encryption Standard (AES) і прискорює шифрування даних у процесорах Intel® Xeon® і процесорах Intel® Core™.

Intel CORE I9-13900K S1700 містить у собі сім нових команд і забезпечує більше швидкий і доступний захист даних і більше високий рівень безпеки, сприяючи поширенню шифрування на нові області IT-середовища.

Шифрування часто рекомендують як кращий спосіб захисту важливих для бізнесу даних, а стандарт AES найбільше широко застосовується для захисту мережного трафіку, особистих даних і корпоративних IT-інфраструктур.

Завдяки останнім досягненням в області хмарних обчислень особиста й важлива для бізнесу інформація виходить за межі традиційних IT-середовищ, тому особливу роль здобувають широко використовувані й безпечні стандарти шифрування, такі як AES, і механізми прискорення, такі як Intel CORE I9-13900K S1700.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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

– Огляд існуючих систем захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

– Дослідження системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

– Програмна реалізація системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700, 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

Призначенням системи є захист інформації з використанням інструкцій процесора Intel CORE I9-13900K S1700.

AES є поточним стандартом шифрування, використовуваним урядом США, і він прийшов на зміну попередньому стандарту, потрійному DES, що використовував стандартний 56-bit ключ. AES може використовувати ключі різної довжини, які характеризуються як AES-128, AES-192 і AES-256. Залежно від довжини ключа може бути до 14 циклів трансформації, необхідної для створення кінцевого зашифрованого тексту.

AES також має кілька режимів роботи:

– electronic codebook (ECB);

– cipher block chaining (CBC);

– counter (CTR);

– cipher feedback (CFB);

– output feedback (OFB).

Ланцюжок зашифрованих блоків (Cipher block chaining) є найпоширенішим режимом, оскільки він надає прийнятний рівень безпеки й не підданий уразливості статистичних атак.

Основною проблемою з такими просунутими методами шифрування, як AES з CBC є те, що вони споживають багато ресурсів процесора. Це особливо стосується серверів, але може створювати проблеми й для завантажених клієнтських систем, оскільки на них установлюються менш потужні процесори. Це означає, що ви можете виявитися перед вибором: більше високий ступінь захисту проти високого рівня продуктивності вашої системи. Ця ситуація може бути настільки проблематичної на стороні сервера, що такі способи обходу цих

проблем, як SSL або IPsec карти розвантаження (карти розвантаження шифрування) використовуються для зниження рівня навантаження на процесор і дозволяють процесору виконувати й іншу роботу крім створення сеансів і шифрування.

Проблема з картами що додаються, полягає в тому, що вони залежать від додатків і можуть не працювати, залежно від того, для яких цілей ви хочете використовувати їх. Нам потрібно загальне рішення, що буде працювати у всіх сценаріях шифрування AES, щоб не потрібно було робити нічого спеціально для розвантаження завдання шифрування центрального процесора. Нам потрібно рішення 'plug and play', що було б убудоване в ОС і материнську плату.

На щастя, стандарт шифрування AES широко застосовується для захисту мережного трафіку, особистих даних і корпоративних IT-інфраструктур і набір Intel CORE I9-13900K S1700 можна використовувати для прискорення AES-шифрування. При використанні таких надійних, доступних і гнучких рішень набір команд Intel CORE I9-13900K S1700 може допомогти підприємству випереджати зростаючі погрози.

1.2 Область застосування

Областю застосування є системи електронного документообігу. Із самого початку, система електронного документообігу розглядалася як перший крок до автоматизації завдань стандартного виробництва, але пізніше охопило більше широкий спектр рішень. Сьогодні електронне діловодство стає нормою.

Давайте розглянемо основні критерії, які допоможуть проаналізувати можливості різних рішень, а саме з погляду технічної реалізації будь-якого завдання системи електронного документообігу:

– Реєстрація на сайті й подальше уведення документів.

– Безпосередня робота з документами.

– Контроль і керування потоками робіт «Workflow».

– Пошук інформації і її аналіз.

– Забезпечення інформаційної безпеки.

– Надійна підтримка документообігу.

– Стандартна система налаштування.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700, 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

BPCord (Business Process Coordinator)

BPCord (Business Process Coordinator) – це web-based продукт нового покоління, побудований на принципах Service Oriented Architecture (SOA), призначений для керування бізнес-процесами й побудови гнучких, масштабованих, процесно-орієнтованих інформаційних систем компаній. BPCord ставиться до класу Human Centric Business Process Management System (BPM), іноді також називаних Human Interaction Management System (HIMS).

BPCord – це надійна платформа для ефективного керування бізнес-процесами, легко впроваджується, швидко інтегрується у вже існуючу IT-інфраструктуру, сприяє підвищенню керованості компанією й забезпечує короткий строк ROI.

Типові результати впровадження Business Process Management System:

– Збільшення ефективності.

– Збільшення якості надаваних сервісів.

– Зменшення часу виконання процесів.

– Контроль витрат.

– Підтримка організаційних змін.

– Раціональне використання IT і інфраструктури.

BPCord v.2.1 підтримує повний життєвий цикл бізнес-процесів (проектування, впровадження, виконання, моніторинг, поліпшення).

BPCord дозволяє автоматизувати виконання бізнес-процесів і надає користувачам засоби керування й моніторингу бізнес-процесів протягом усього життєвого циклу.

Ключові можливості:

– 100% web-based інтерфейс;

– єдиний інформаційний простір керування бізнес-процесами;

– зручний і простий графічний інтерфейс за принципом єдиної точки входу;

– засобу опису й керування бізнес-процесами;

– налаштування й автоматичний контроль бізнес-правил;

– засобу комунікації (система оповіщення, внутрішні повідомлення);

– керування документами: структурованими (XML) і неструктурованими (Word, Excel, PDF і т.д.);

– можливість одночасної роботи декількох користувачів над одним документом;

– потужний інструмент моніторингу реалізації бізнес-процесів;

– засоби для інтеграції систем, використовуваних для виконання різних бізнес-процесів.

Інструменти й концепції

BPCord – це потужний і легкий в освоєнні набір інструментів для проектування бізнес-процесів і організації ефективної комунікації співробітників у ході їхнього виконання:

– Підсистема BPCord Portal забезпечує єдину точку входу й гнучкі уніфіковані інтерфейси кінцевих користувачів (на базі концепції BPCord Unified User Interface) для ініціації, керування й виконання бізнес-процесів і завдань компанії.

– Підсистема керування бізнес-процесами й завданнями:

BPCord Human Process Model забезпечує:

– Засоби проектування схем процесів з можливістю використання найбільш підходящої моделі. У розпорядженні два візуальних дизайнери: Workflow Designer і Project Designer.

– Інструмент конструювання як завгодно складних процесів на базі BPCord Composite Process Model.

– Реалізацію концепції Human Process Life Cycle, що дає можливість управляти змінами в процесах і надає кошти комунікації менеджерів і виконавців у ході робіт над екземпляром процесу.

BPCord Human Task Model забезпечує повноцінне керування життєвим циклом Human Task і надає потужні кошти комунікацій у ході виконання завдань.

BPCord Multi-Matrix забезпечує функціонування матричної системи керування:

Підтримує:

– функціональну організаційну структуру й розмежування посадових повноважень співробітників;

– автоматичне формування процесних структур – цільових команд співробітників (виконавців процесів);

– ієрархію цілей і напрямків діяльності компанії.

Контролює взаємодія функціонального й процесного потоку керування.

Підсистема BPCord Documents, побудована на базі концепції BPCord Task Centric Application, дозволяє працювати як з неструктурованими (Word, Excel, PDF і т.д.), так і зі структурованими (XML) документами. Широкий набір функцій забезпечує керування:

– життєвим циклом документів;

– версіями документів;

– шаблонами документів;

– формами уведення структурованих документів;

– класифікацією документів.

BPCord Integration Layer забезпечує можливість підключення зовнішніх функціональних модулів і інтеграцію систем уже, що експлуатуються в компанії з використанням механізмів, Web Services.

BPCord Business Processes Library дозволяє структурувати, зберігати й використовувати спроектовані відповідно до галузевих стандартів (CMMI, eTOM, і т.д.) бібліотеки схем процесів, які можуть бути задіяні при запуску process-centric рішення у вашій компанії.

BPCord забезпечує повноцінну реалізацію process centric підходу керування компанією на базі концепції BPCord CPM (Composite Process Model), надаючи можливість використання декількох моделей бізнес-процесів, які можуть бути задіяні залежно від вимог до керування конкретним процесом.

Більшість систем пропонують єдину модель бізнес-процесу, яку можна назвати «алгоритмічна модель». У цій моделі бізнес-процес описується (або спеціальною мовою, або за допомогою графічної нотації) як детермінована послідовність завдань і паттернів керування. При цьому, саме керування ходом виконання такого алгоритму покладає на спеціалізований автомат (Workflow Engine).

Безумовно, алгоритмічна модель гарна для керування устояними, регулярно повторюваними процесами, на які вплив зовнішніх факторів або відсутній, або вкрай незначно. У цьому випадку бізнес-аналітик дійсно може передбачити всі можливі варіанти розвитку подій і спроектувати схему бізнес-процесу (алгоритму) таким чином, щоб процес адекватно реагував на кожну можливу ситуацію.

На жаль, у реальному житті все значно складніше, і спроба описати всі процеси компанії на базі алгоритмічної моделі дуже часто закінчується невдачею.

Очевидно, що як мінімум, всі проектні процеси в алгоритмічну модель не укладаються. Не укладаються в неї й процеси, які, як правило, розвиваються по стандартному алгоритмі, але у випадку непередбачених подій починають

управлятися людиною. Разові доручення співробітникам від керівників і персональних завдань також не реалізовані в алгоритмічній моделі.

Таким чином, необхідно або намагатися побудувати супер-модель, що буде покривати всі мислимі варіанти процесів і при цьому одержати «монстра», працювати з яким зможуть тільки висококласні IT-фахівці, або визнати очевидність того, що наявність у бізнес-процесі людського фактора приводить до необхідності використання різних моделей процесів.

BPCord Composite Process Model надає в розпорядження бізнес-аналітиків і менеджерів дві базові моделі процесів: Workflow і Project. BPCord CPM дозволяє використовувати комбінації двох базових моделей для забезпечення ефективного керування складними процесами.

BPCord CPM також містить у собі спеціальні засоби й для керування разовими дорученнями (One-time Human Tasks) і особистими завданнями (Personal Tasks).

BPCord Multi-Matrix – це концепція, що забезпечує реалізацію process centric підходу, органічно включаючи в нього колектив компанії й відносини співробітників усередині колективу. BPCord підтримує матричну структуру, що визначається двома потоками керування:

– Функціональний потік керування – заснований на ієрархічному підпорядкуванні співробітників відповідно до професійної спеціалізації. Основне завдання функціонального потоку – виконання функції. Забезпечення максимально ефективного виконання завдань у рамках професійної спеціалізації.

– Процесний (цільовий) потік керування – заснований на підпорядкуванні співробітників відповідальним менеджерам процесів. Основне завдання процесного потоку – досягнення мети. Забезпечення оптимальних схем процесів і ефективне керування ходом виконання екземплярів процесів.

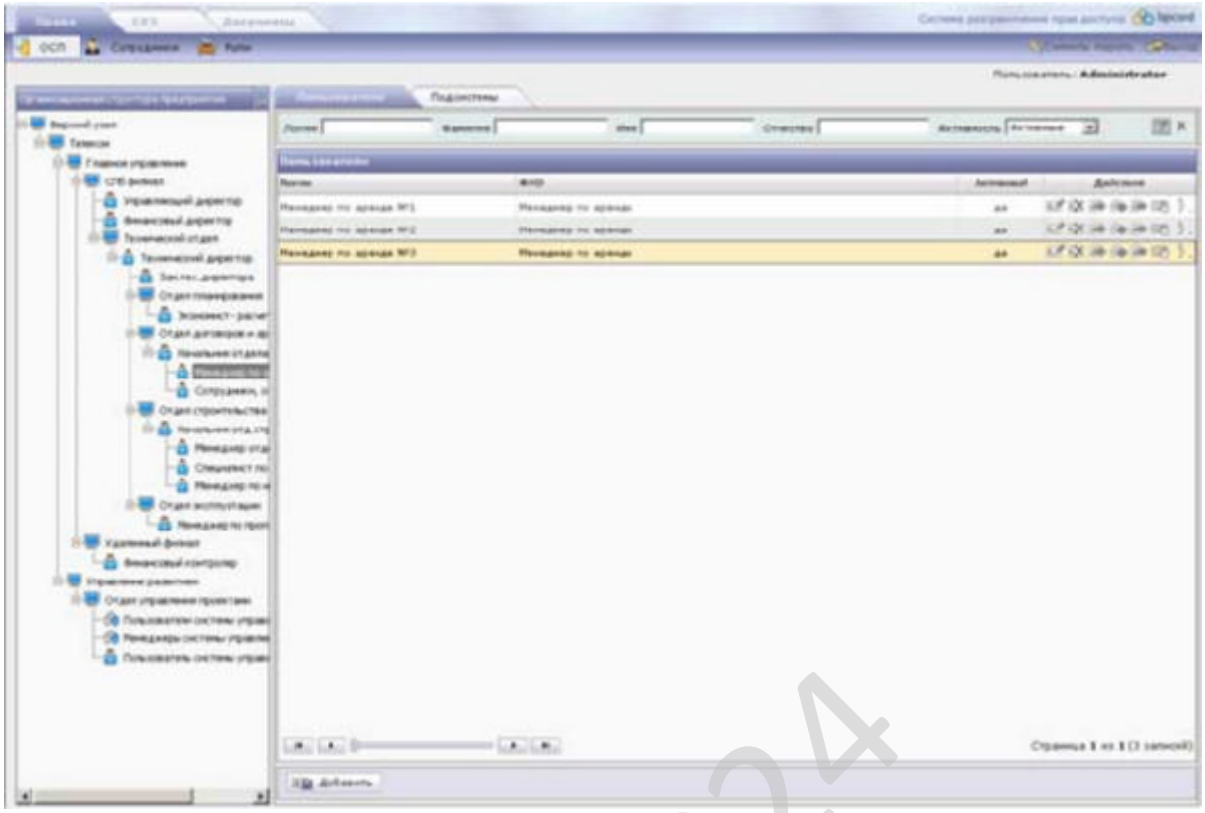

Рисунок 2.1 – Інтерфейс користувача BPCord

Архітектура системи

BPCord, побудований на базі технологій SOA і Web Services, забезпечує унікальні можливості інтеграції з інформаційними системами, що експлуатуються в компанії, і може бути задіяний як центральний компонент побудови єдиного інформаційного простору.

BPCord є Web-based продуктом, розробленим на платформі Microsoft.NET. Всі функції, необхідні користувачам для виконання бізнес-процесів і завдань доступні через інтерфейс Web-браузера. Це сильно прискорює розгортання рішення й серйозно скорочує витрати на експлуатацію, тим самим істотно знижуючи сукупну вартість володіння системою.

BPCord спроектований із застосуванням концепцій stateless архітектури, що дозволяє будувати кластерні рішення для систем, що вимагають підвищеної надійності функціонування й адаптованих до великих пікових навантажень.

Інтеграція із продуктами Microsoft

Сьогодні MS Project є найбільш популярним продуктом для керування проектами. Використовувані технології й засоби BPCord дозволяють описувати як складні бізнес-процеси, так і окремі завдання для виконання разових доручень. Інтеграція з MS Office Project 2007 робить BPCord унікальним по своїх функціональних можливостях рішенням – єдиним порталом, здатним управляти бізнес-процесами будь-якого типу й складності. Спільне використання BPCord і MS Office Project 2007 дозволяє компанії управляти бізнес-процесами максимально ефективно

Методологія впровадження BPCord Smart Start

З методологією швидкого старту BPCord Smart Start можливо одержати результат практично негайно: послідовність дій проста, не вимагає спеціальної технічної підготовки й займає кілька днів. можливо швидко запустити пілотний проект і одержати готову до експлуатації систему вже через пару тижнів. Все це дозволить вам зменшити витрати на проект і забезпечить короткий строк ROI.

Ефект від впровадження

BPCord буде працювати в режимі реального часу, це дозволить вам:

– більш оперативно приймати управлінські рішення;

– стандартизувати рутинні операції;

– зробити більше чітким взаємодія всіх підрозділів компанії й співробітників, що приймають участь у різних бізнес-процесах.

Ефект від впровадження системи BPCord може бути досягнуть у наступних областях:

– підвищення керованості компанією;

– керованість бізнес-процесами протягом усього життєвого циклу;

– побудова ефективної стратегії керування людськими ресурсами;

– збільшення продуктивності за рахунок координації керування роботами;

– скорочення часу виконання завдань бізнесу-процесу.

OfficeMedia

Система OfficeMedia призначена для автоматизації документообігу невеликих організацій, що не мають віддалених офісів, що використовують один сервер і локальну комп'ютерну мережу з невеликою кількістю робочих місць. За роки свого існування система заслужила визнання в підприємств всіляких сфер діяльності, загальне число користувачів системи перевищує 20 000 чоловік.

Система OfficeMedia вирішує наступні завдання:

– Координація роботи, колективне й індивідуальне планування робочого дня.

– Швидкий доступ до інформації про зовнішні організації й про поточний стан відносин з ними.

– Надання засобів для ділової переписки.

– Реєстрація документів, що циркулюють в організації.

– Можливість оперативної роздачі доручень.

– Аналіз і контроль виконавської дисципліни.

– Узгодження, ознайомлення, виконання документів.

Система OfficeMedia складається із чотирьох комплектів:

«Діловодство»

Комплект призначений для автоматизації роботи секретарів або відділу діловодства. Складається з наступних баз даних:

– «Реєстрація документів» – призначена для організації систематичного обліку всіх вхідних, вихідних і внутрішніх документів організації на основі регістраційно-контрольних карток. У базі реалізовані механізми контролю за виконанням документів, резолюціями керівництва, виконавської дисципліни співробітників організації й т.п. База даних забезпечує автоматичну підготовку різних звітів і стандартних облікових документів для служб документаційного забезпечення керування.

– «Бібліотека документів» – служить для підготовки документів довільного формату, їхньої класифікації по темах (електронним папкам),

контекстного пошуку інформації й друку документів. База даних може бути використана для зберігання документів довільного змісту, таких як нормативні документи організації й довідкових матеріалів, для підготовки проектів документів і т.д.

– «Узгодження» – призначена для відправлення документів на узгодження й для зберігання результатів узгодження. Дозволяє здійснювати автоматичне розсилання документів співробітникам організації по внутрішній пошті для електронного візування. Здійснюється оперативний контроль процесу узгодження.

– «Ознайомлення» – призначена для відправлення документів на ознайомлення. Дозволяє здійснювати автоматичне розсилання документів співробітникам організації по внутрішній пошті для ознайомлення. Здійснюється оперативний контроль процесу ознайомлення.

– «Звернення громадян» – призначена для ведення діловодства на підставі звернень фізичних осіб.

– «Організаційно-розпорядницькі документи» – призначена для формування проектів наказів, розпоряджень, службових записок, їхнього узгодження, підпису й контролю виконання.

Бази даних комплекту можуть використовуватися як незалежно, так і разом, в останньому випадку забезпечуючи автоматичну маршрутизацію документів усередині системи. Ви самі вибираєте ті бази, з якими будете працювати.

«Зовнішні контакти»

Комплект дозволяє автоматизувати роботу співробітників, відповідальних за підтримку зовнішніх контактів організації й роботу із клієнтами. Бази даних комплекту дозволяють накопичувати інформацію про потенційних клієнтів, вести роботу з партнерами, постачальниками й т.д. Комплект дає можливість організувати роботу відділу продажів, планувати й контролювати роботу із проектів.

За допомогою даного комплекту можливо:

– інформувати співробітників про розвиток відносин із зовнішніми організаціями, про досягнуті домовленості й подальші плани робіт;

– фіксувати інформацію про співробітників зовнішніх компаній, про спільні проекти;

– планувати заходи й зустрічі;

– заносити інформацію про результати переговорів, фіксувати телефонні дзвінки і їхні тематики;

– фіксувати вхідну й вихідну кореспонденцію.

Комплект складається з наступних баз даних:

– «Зовнішні контакти» – призначена для зберігання інформації про організації, що є партнерами, постачальниками послуг, клієнтами й т.п. До функцій бази даних ставляться: розсилання матеріалів через поштові й електронні канали зв'язку, можливість зберігання адрес, телефонів, інформації про контактні особи, характеристик пропонованих послуг, історії взаємин.

– «Проекти» – призначена для планування, підготовки й проведення проектів в організації.

– «Розсилання» – дозволяє зберігати й становити списки розсилання, готовити й розсилати документи, рекламні матеріали, новини й т.д.

– «Секретар» – дозволяє вносити інформацію про перші контакти, реєструвати телефонні дзвінки й кореспонденцію, що надходять в організацію, автоматично повідомляти про їх співробітників, відповідальних за роботу із клієнтами й підтримку зовнішніх контактів організації.

«Керування й планування»

Комплект призначений для автоматизації роботи керівників проектів, відділів IT і внутрішньої технічної підтримки, а також служб, відповідальних за підтримку організаційно-розпорядницької діяльності підприємства. Бази даних цього комплекту призначені для спрощення роботи з договорами, контролю за дорученнями й зверненнями внутрішніх служб у відділ технічної підтримки.

За допомогою даного комплекту можливо:

– докладно відслідковувати всі стадії робіт за договорами;

– працювати з дорученнями;

– автоматизувати процеси рішення технічних проблем, що виникають у користувачів.

Комплект складається з наступних прикладних баз даних:

– «Доручення» – призначена для реєстрації й контролю виконання завдань, не прив'язаних до яких-небудь документів.

– «HelpDesk» – призначена для інформаційної підтримки процедур збору, узагальнення й ведення архіву усунутих проблем з метою їхнього подальшого аналізу й використання результатів при виникненні аналогічних ситуацій, обліку навантаження на співробітників технічної служби, аналізу стану виконавської дисципліни при розгляді заявок.

– «Договори» – підготовка проектів договорів, планування робіт за договорами, контроль виконання договорів.

«Облік матеріальних цінностей»

Комплект вирішує наступні завдання:

– облік матеріальних цінностей, що перебувають на балансі організації;

– автоматизація процесу формування заявок на одержання збережених у підсистемі об'єктів;

– автоматизація процесу прийому – передачі матеріальних цінностей співробітникові, на склад, у резерв і т.д.;

– контроль переміщення матеріальних цінностей усередині організації.

Комплект складається з наступних баз даних:

– «Довідник УМЦ».

– «Архів УМЦ».

– «Облік устаткування».

Всі ці комплекти утворять єдину, налагоджену систему, що автоматизує роботу організації.

Загальні відомості про архітектуру

Реалізація на Lotus Domino/Notes – кращої документоорієнтованій платформі.

Система електронного документообігу CompanyMedia схожа на «листковий пиріг», де кожний окремий рівень виконує певні функції й здійснює взаємодію й обмін даними з іншими:

1. В основі – платформа Lotus Domino/Notes, що є базисом, забезпечує одержання системних повідомлень через пошту користувача, надає календарне планування, здійснює функції захисту системи від злому й можливості інтеграції із зовнішніми додатками;

2. Наступний рівень – ядро системи ActiveFrame, що надає можливості роботи з корпоративними довідниками й класифікаторами, обслуговує внутрішні сервіси системи, здійснює адміністрування й керування прикладними базами даних і комунікації між організаціями;

3. Сама система CompanyMedia представляє із себе набір баз даних, сформованих по типах документів, збережених у ній (журнали обліку документів).

4. Користувачі працюють із системою через єдиний інтерфейс, «точку входу» у систему – СМ-універсальне робоче місце, що дає доступ до документів системи відповідно до наданих прав, і дозволяє працювати як у локальній мережі, так і віддалено, з використанням веб-інтерфейсу й доступу через Інтернет.

CORPORATE BUSINESS

«Система оперативного керування CORPORATE BUSINESS» призначена для комплексної, наскрізної автоматизації: документообігу, взаємодії співробітників, взаємодії із клієнтами й бюджетування в комерційних організаціях.

Система CORPORATE BUSINESS увібрала в себе технологічні інновації, які виводять систему в лідери програмного забезпечення в усіх напрямках.

Основні переваги системи CORPORATE BUSINESS:

– ергономічний сучасний інтерфейс побудований за принципом «Візуального шару». Ергономічний сучасний інтерфейс «Системи оперативного керування» побудований за принципом «Візуального шару» – вся необхідна користувачеві інформація представлена «як на долоні». Ергономіка в «Системі оперативного керування» – це інтуїтивно-зрозумілий дружній інтерфейс для приємної роботи – кожна дія користувача в системі спрямовано на максимальне скорочення часу для одержання потрібної інформації або подання цієї інформації в необхідному розрізі. Інтерфейс системи програмується таким чином, що б, у першу чергу, із Системою було приємно працювати самим вимогливим користувачам. «Візуальні шари» дозволяють кожному користувачеві в будь-якому підрозділі приділяти мінімум часу на уведення, обробку інформації й формування звітності;

– інтеграція з текстовими й табличними редакторами з пакета Microsoft Office. При вході в «Систему оперативного керування» кожний користувач може задіяти Windows-Авторизацію – це дозволяє Системі «запам'ятати» користувача й не просити вводити пароль при кожному вході в Систему. Тісна інтеграція з Microsoft Office обумовлена розповсюдженістю й популярністю таких редакторів як WORD і EXCEL, які активно використовуються багатьма співробітниками при підготовці електронних редакцій документів. Щільна інтеграція з пакетом Microsoft Office дозволяє користувачам «Системи оперативного керування» затрачати мінімум часу на підготовку документів. Наприклад, всі дані з «Системи оперативного керування» автоматично можуть бути передані у файли формату Microsoft WORD, а тісна інтеграція з Microsoft EXCEL дозволяє будувати будь-які статистичні звіти по натисканню однієї кнопки миші;

– маршрутизація й зберігання електронних версій документів, використання ЕЦП;

– масштабованість. «Система оперативного керування» гарантує збереження інвестицій в інформацію й ПЗ при переході на могутнішу апаратну

платформу. Система адаптована до розширення пропонованих вимог і зростанню обсягів розв'язуваних завдань, обсягу оброблюваних даних і кількості інтенсивно працюючих користувачів. Для оптимізації роботи з даними, що б обсяг архівних даних не робив впливу на швидкість роботи з оперативними даними, кожна архівна база даних ведеться окремо в розрізі одного року. При збільшенні обсягу оброблюваної інформації, у користувачів «Системи оперативного керування» є можливість регулювання доступу до архівних баз даних. Технологія використання інтегрованого клієнт-сервера дозволяє рівномірно розподіляти навантаження між сервером і робочими станціями користувачів;

– динамічне відображення інформації в режимі real-time у різних зрізах;

– наскрізна обробка інформації. «Система оперативного керування» дозволяє один раз уведену інформацію одним підрозділом, успішно обробляти цю інформацію у всіх інших підрозділах. Наскрізний облік інформації гарантує можливість одноразової обробки даних у різних підрозділах, наприклад: у відділі по роботі із клієнтами, у юридичному відділі, у бухгалтерії й в адміністрації. «Система оперативного керування» – це одна з перших автоматизованих систем даного класу, де був застосований принцип «наскрізної обробки інформації» (STP – Straight Through Processing). Наскрізна обробка інформації дозволяє використовувати єдиний програмний продукт і заощадити фінансові засоби на придбання й супровід розрізнених програмних продуктів;

– автоматичне розсилання поштових і системних повідомлень користувачам.

Візуальний шар docflow

Електронний документообіг – це ефективне:

– керування підприємством;

– керування організацією;

– керування компанією;

– керування справами.

Система документообігу – це новий якісний рівень розвитку управлінських функцій.

Візуальний шар bb docflow органічно взаємозалежний з іншими шарами, які, взаємно доповнюючи один одного, створюють комплексну збалансовану систему по оперативному керуванню організацією, перекриваючи по своїй значимості й змісту звичайну систему електронного документообігу.

У Візуальному шарі bb docflow документообіг трансформується під кожного конкретного користувача, дозволяючи працювати з документами з різних точок зору, залежно від того, що цікавить користувача в кожній конкретній ситуації: у розрізі типів документів, номенклатури справ і проектів, у розрізі клієнтів і контрагентів, у розрізі статусів документів і завдань, у розрізі оргструктури організації.

Кожний електронний документ характеризується своєю «Обкладинкою» (карткою) документа. В «Обкладинку» заноситься інформація, по якій автоматично формується сам документ по заданому шаблоні. Можливий варіант простого прикріплення кожного файл-документа або уведення документа безпосередньо зі сканера.

На «Обкладинці» документа може бути заданий «гнучкий» або «твердий» маршрут обробки документа із вказівкою іменних або рольових підписів, а так само формується список на розсилання документа для ознайомлення. Всі дії користувачів протоколюються. Якщо необхідно вказується документ-підстава, взаємозв'язок даного нового документа з іншими документами, формування Завдань (доручень) по документу.

Будь-який документ або завдання (доручення) можна поставити на контроль із вказівкою Куратора, конкретного строку виконання.

Підтримано можливість ведення секретного діловодства.

Реалізовано убудований контекстний пошук по файлах документів за спеціальною технологією, оптимізованої для роботи з кирилицею.

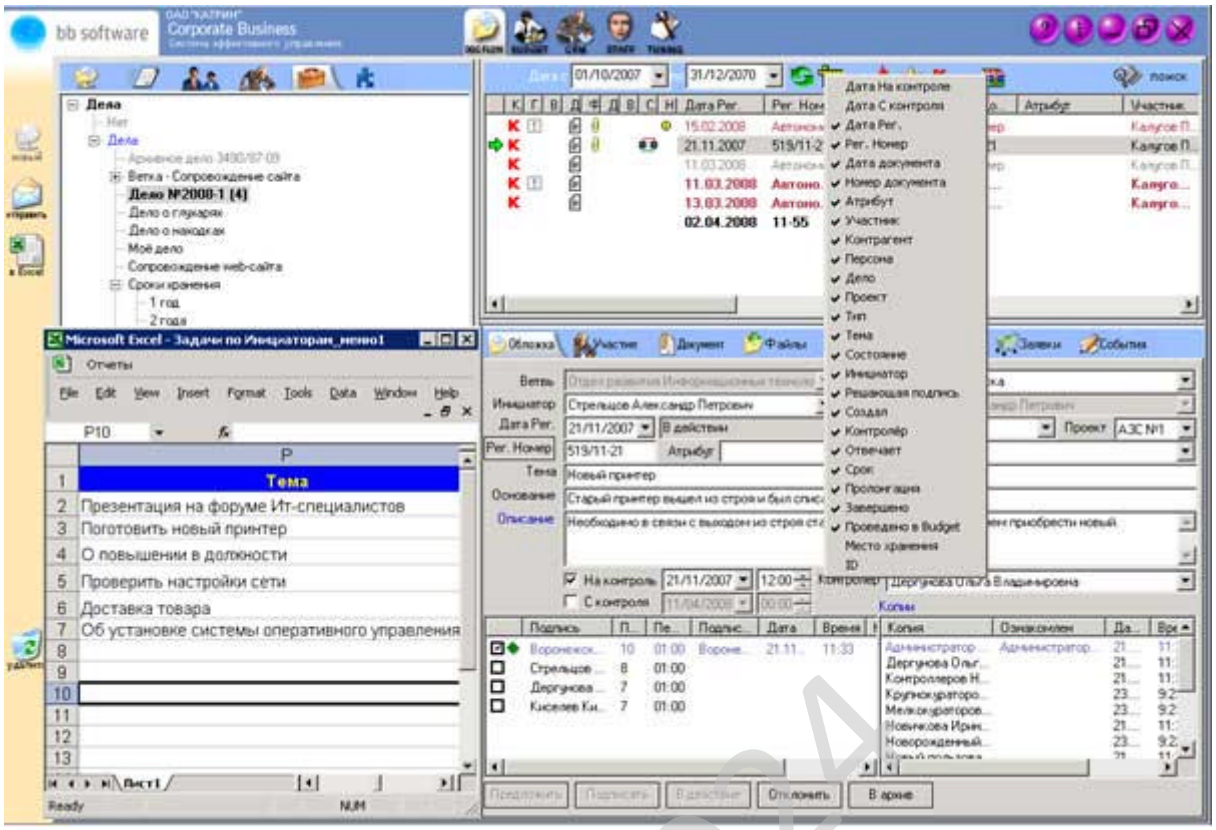

Рисунок 2.2 – Інтерфейс користувача CORPORATE BUSINESS

Візуальний шар bb docflow призначений для виконання наступних функцій:

– Створення/редагування/видалення обкладинок (облікових карток) документів і завдань (доручень).

– Створення/редагування/видалення документів у форматі Microsoft Office.

– Підтримка використання шаблонів документів у форматі Microsoft Office.

– Колективна робота з файлами-документами.

– Багатостадійна обробка документів і завдань.

– Підтримка різних форматів файлів *.doc, *.xlt, *.rar, *.zip, *.html, графічних (сканер, факс) і аудіо-відео файлів.

– Створення дочірніх (залежних) завдань/ доручень /резолюцій.

– Підтримка зв'язків облікових карток у всіх можливих комбінаціях: документ-документ-..., документ-підстава-..., документ-завдання-....

– Зручна й швидка навігація по всіх зв'язках.

– Відправлення файл-документів по електронній пошті, незалежно від поштового клієнта.

– Автоматичне розсилання повідомлень користувачам системи, у тому числі й по електронній пошті.

– Підготовка й печатка звітів із системи з використанням пакета Microsoft Office: Word and Excel.

– Контекстний пошук по будь-яких полях і змісту документів.

– Установка списку що підписують / що погодять і пріоритету накладення підпису /установка списку розсилання.

– Збереження проміжних версій документів (збереження історії роботи з документами).

– Протоколювання часу: Накладення підпису /Відхилення; Створення /Редагування; Розсилання копій /Ознайомлення з копією; Видачі доручення /Виконання доручення.

– Використання зручного «рольового» адміністрування.

– Використання різних носіїв інформації для зберігання «Авторського ключа» і закритого ключа засобу криптографічного захисту інформації.

– Зручне подання документів і завдань у розрізі: Типів документів, Справ і Проектів, Станів документів, Структури організації й Клієнтів (при використанні Візуального шару CRM).

– Можливість налаштування відображення колонок реєстру.

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 захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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

Розроблювальне програмне забезпечення, назвемо його умовно «Захищений електронний документообіг Intel CORE I9-13900K S1700» являє собою комплекс систем, що автоматизують діловодство, документообіг і бізнес-процеси.

Система «Захищений електронний документообіг Intel CORE I9-13900K S1700» – сімейство систем електронного документообігу, автоматизації бізнес-процесів, інформаційної підтримки організаційно-розпорядницької й виробничо-господарської діяльності в організаціях будь-якого розміру.

Не всі завдання роботи з документами можуть бути вирішені в рамках системи автоматизації загального діловодства. Наявність окремих систем, що автоматизують різні бізнес-процеси (керування персоналом, робота із клієнтами й ін.), а також роботу з різними видами документів (договорами, зверненнями громадян, заявками та ін.) дозволяє організувати бізнес-процеси в повній відповідності зі стандартом і вирішувати відповідні завдання на високому професійному рівні. При цьому можна впровадити в компанії як одну, так і кілька систем «Захищений електронний документообіг Intel CORE I9-13900K S1700» у будь-якій комбінації залежно від того, які процеси необхідно автоматизувати.

«Захищений електронний документообіг Intel CORE I9-13900K S1700» забезпечує користувачам такі основні можливості, як:

– підготовка й узгодження документів;

– реєстрація всіх видів документів компанії, у тому числі вхідної й вихідної кореспонденції, внутрішніх документів, організаційно-розпорядницьких документів та ін.;

– створення й ведення папок документів;

– створення по всіх видах документів доручень, резолюцій і можливість реєстрації відповідей на них;

– контроль виконання документів і резолюцій;

– сканування великої кількості документів, створення й ведення електронних архівів документів;

– підготовка, реєстрація договорів, узгодження, візування, зберігання їхнього актуального реєстру;

– інформаційна підтримка роботи із клієнтами;

– планування робіт, контроль виконання планів;

– ведення проектів;

– облік робочого часу співробітників;

– підготовка нарад і засідань, контроль виконання рішень;

– автоматизація кадрового діловодства;

– автоматизація взаємодії користувачів і служби технічної підтримки.

Можливість здійснювати в системі «Захищений електронний документообіг Intel CORE I9-13900K S1700» весь цикл роботи з документами (від їхнього створення/одержання до архівування) приводить до скорочення часу на їхнє проходження по структурних підрозділах, пошук документів, необхідних керівництву для прийняття управлінських рішень, що підвищує їхню якість і надійність за рахунок повноти й своєчасності надаваної інформації. Також із впровадженням системи «Захищений електронний документообіг Intel CORE I9-13900K S1700» скорочуються невиробничі витрати (пошук, повторне узгодження документів, дублювання інформації й т.д.) і витрати на придбання видаткових матеріалів (паперу, копіювально-множного встаткування), зміст паперового архіву.

«Захищений електронний документообіг Intel CORE I9-13900K S1700» характеризується рядом вигідних переваг:

– готовність до роботи, тобто відсутність необхідності розробляти або моделювати систему на основі конструктора забезпечує можливість почати

використовувати її відразу, скоротивши ризики на впровадження системи й заощадивши на цьому час і матеріальні засоби;

– модульність системи «Захищений електронний документообіг Intel CORE I9-13900K S1700», тобто наявність у її составі декількох самостійних і одночасно взаємозалежних систем, що автоматизують різні процеси, дозволяє вибрати потрібні, і автоматизувати саме ті процеси, які мають потребу в цьому; багатофункціональність, тобто можливість автоматизувати не тільки діловодство, але й багато інших процесів (а до складу системи «Захищений електронний документообіг Intel CORE I9-13900K S1700» входить більше 10 систем, розрахованих на різні завдання) дозволяє розширювати границі автоматизації;

– можливість налаштувати або доробити систему так, щоб вона автоматизувала процеси, які можуть бути властиві саме конкретної організації, що дозволяє враховувати специфіку бізнесу компанії, а також надалі адаптувати її під мінливі вимоги самостійно, заощаджуючи час і гроші;

– масштабованість (можливість надалі, після впровадження системи, підключити до неї інші системи «Захищений електронний документообіг Intel CORE I9-13900K S1700», автоматизувати нові робочі місця) дає можливість у потрібний термін автоматизувати необхідні бізнес-процеси, підключити до роботи в системі потрібних співробітників;

– корпоративність, тобто здатність системи працювати в компаніях зі складною (кілька відділів, департаментів і т.п.), у тому числі територіально-розподіленою структурою (кілька філій, дочірніх компаній, розташованих у різних містах у різних годинних поясах), дозволяє підключити до роботи в ній співробітників основний, дочірніх і залежних компаній. За рахунок підключення до роботи в системі всіх співробітників компанії (як у головному офісі, так і у філіях і дочірніх компаніях), що працюють із документами, строгого контролю дотримання ними посадових обов'язків, регламентів і процедур, прийнятих у компанії підвищується прозорість її документообігу й ділових процесів, прискорюються інформаційні потоки, що підвищує керованість компанії;

– реалізована в системі можливість ставити виконання документів і доручень по них на контроль, тобто, прописувати, у які строки документ і доручення по ньому повинні бути виконані, хто повинен брати участь у їхньому виконанні, а також можливість відстежити, у кого документ перебуває, на який він стадії виконання (проглядається й т.д.) приводить до спрощення одержання інформації про поточний стан документа, підвищенню виконавської дисципліни;

– можливість розмежувати право доступу до інформації на підставі посадових обов'язків співробітника забезпечує захищеність документообігу від несанкціонованого доступу, тим самим ліквідується можливість витоку інформації;

– наявність механізмів створення юридично значимого електронного документообігу дозволяє перейти на повністю безпаперовий документообіг;

– наявність декількох типів робочих місць (для менеджерів і керівників, що працюють в основному зі змістом документа, предметних фахівців, що працюють, як правило, з реквізитами документів), що відбивають специфіку роботи з інформацією залежно від посадових обов'язків співробітників, тобто відбиття в системі саме тих функціональних можливостей, які необхідні співробітникам для виконання професійних обов'язків, робить роботу в системі «Захищений електронний документообіг Intel CORE I9-13900K S1700» комфортною для фахівців різних категорій;

– сумісність системи з різними операційними системами (Sun Solaris, Linux, Windows) дає можливість працювати із установленої в компанії операційною системою, і не купувати нову, заощаджуючи тим самим засобу на її придбання, а також не бути прив'язаним до конкретного постачальника встаткування;

– інтегруємість системи «Захищений електронний документообіг Intel CORE I9-13900K S1700» з іншими, уже використовуваними на підприємстві системами, дозволяє взаємодіяти ними, використовувати інформацію, що зберігається в них;

– можливість роботи через Web-браузер забезпечує миттєвий доступ до документів, що зберігається в системі «Захищений електронний документообіг Intel CORE I9-13900K S1700», навіть якщо в співробітника вона локально не встановлена (наприклад, якщо він у відрядженні, у відпустці), що дозволяє йому завжди оперативно відпрацьовувати вступників документи й доручення по них;

– можливість підтримки в системі єдиного стандарту роботи з електронними документами забезпечує уніфікацію, формалізацію й строгу регламентованість технологій діловодства, документообігу й бізнес-процесів;

– надійність, висока «відказостійкість» системи дозволяє їй працювати практично 24 години на добу, 365 днів у році.

Експлуатація системи «Захищений електронний документообіг Intel CORE I9-13900K S1700» дозволяє виключити ризики:

– несвоєчасної доставки інформації;

– невиконання або несвоєчасного виконання співробітниками виданих керівництвом доручень;

– втрати інформації при її передачі й зберіганні;

– тривалого узгодження проектів документів, і, відповідно, неприпустимо повільного реагування на зміни зовнішньої бізнес-середовища;

– несвоєчасного виконання ділових зобов'язань.

Система «Захищений електронний документообіг Intel CORE I9-13900K S1700» може функціонувати у великих компаніях і корпораціях, які мають розгалужену систему філій у містах, розташованих у різних годинних поясах. Це дозволяє створити єдиний інформаційний простір між головним офісом і філіями компанії.

У системі «Захищений електронний документообіг Intel CORE I9-13900K S1700» можуть бути використані технології, що забезпечують захищений, юридично значимий електронний документообіг. Зокрема, система Locker. Вона виконує операції шифрування й обчислення електронно-цифрового підпису на основі сертифікованих засобів крипто захисту, які реалізуються з використанням інструкцій процесору Intel CORE I9-13900K S1700 (СКЗІ). Із системою також прекрасно працює eToken – електронний ключ-брелок, що дозволяє відмовитися

від використання жорсткого диска й дискет для зберігання пароля доступу в систему, забезпечує високу захищеність інформації й дає можливість легко працювати із системою з різних робочих місць.

«Захищений електронний документообіг Intel CORE I9-13900K S1700» створювався в строгій відповідності з державними стандартами в області діловодства й відповідає вимогам діючої державної нормативної бази.

Принцип роботи Intel CORE I9-13900K S1700

Завдяки апаратній реалізації деяких елементів алгоритму AES набір Intel CORE I9-13900K S1700 прискорює виконання AES-додатків і робить їх більше захищеними.

Нові команди із набору Intel CORE I9-13900K S1700 прискорюють шифрування й дешифрування, а також поліпшують генерацію ключів і обробку таблиць, надаючи підтримку при нескладному множенні.

У результаті мінімізуються вимоги до продуктивності додатка при традиційній криптографічній обробці й забезпечується підвищена безпека завдяки усуненню атак на AES через бічний канал, пов'язаних із традиційними програмними методами пошуку в таблицях.

Відмінне сполучення: технологія Intel® Data Protection з набором інструкцій AES-NI і шифруванням Secure Key

Шифрування – це основа захисту даних. А базою для шифрування є:

1. Надійні алгоритми, наприклад, AES.

2. Надійні ключі, наприклад, надійний набір з випадкових чисел.

Новий набір інструкцій Intel CORE I9-13900K S1700, що у цей час доступний у процесорах серії Intel Xeon 5600, відповідає цим критеріям. Раніше цей процесор був відомий під своєю кодовою назвою Westmere-EP. AES-NI виконує деякі кроки AES на апаратному рівні, прямо в мікросхемі процесора. Однак ви повинні знати, що AES-NI на процесорі не включає повний процес реалізації AES, лише деякі компоненти, необхідні для оптимізації продуктивності. AES-NI робить це шляхом додавання шести нових AES інструкцій: чотири з них для шифрування/ розшифровки, одна для колонки 'mix'

(змішування), і одна для генерування тексту наступного циклу 'next round' (де кількість циклів контролюється довжиною біт, обраних вами).

Одним із чудових моментів в Intel CORE I9-13900K S1700 полягає те, що, оскільки він побудована на базі апаратних засобів, немає необхідності зберігати в пам'яті таблиці перегляду, а блоки шифрування виконуються в процесорі. Це знижує шанси успішності атак сторонніх каналів ('side channel attacks'). До того ж, Intel CORE I9-13900K S1700 дозволяє системі виконувати ключі більшої довжини, у результаті чого дані більш надійно захищені.

На справжній момент Intel CORE I9-13900K S1700 концентрується в основному на трьох моментах:

– Захищені транзакції через інтернет і в інтрамережі.

– Повне шифрування диска (наприклад, як у випадку з Microsoft BitLocker).

– Шифрування прикладного рівня (частина захищеної транзакції).

Захищені транзакції по інтернету й інтрамережі можуть включати використання SSL для підключення до захищеного веб сайту в інтрамережі або інтернеті. До того ж, IPsec тунельний і транспортний режим користуються все більшою популярністю для захисту сеансів в інтрамережі, а у випадку з DirectAccess в інтернеті. Варто враховувати, що SSL використовується для захисту комунікацій рівнів 7, а IPsec використовується для захисту комунікацій мережного (третього) рівня.

Останнім часом можна було чути, що комп'ютерна хмара стає наступним великим проривом у комп'ютерному світі, і постачальники послуг комп'ютерної хмари значно виграють від Intel CORE I9-13900K S1700, де більшість їхніх комунікацій буде здійснюватися через зашифрований канал. Що стосується IPsec, якщо із сервером є всього трохи IPsec з'єднань, то буде цілком достатньо й розвантаження SSL. Але якщо ваш сервер завантажений, Intel CORE I9-13900K S1700 окремо або в сполученні з SSL розвантаженням буде більше підходящим рішенням.

До того ж, тут є компонент транзакції ('secure transactions'). До того ж до шифрування прикладного або мережного рівня, є шифрування прикладного рівня, що використовує Intel CORE I9-13900K S1700. Наприклад:

– Бази даних можна шифрувати.

– Пошту можна шифрувати.

– Служби керування правами використовують шифрування.

– Сама файлова система може бути зашифрована (на відміну від шифрування на дисковому рівні).

Такі додатки, як Microsoft SQL можуть використовувати шифрування Transparent Data Encryption (TDE) для автоматичного шифрування записів, внесених у базу даних.

В остаточному підсумку виходить, що Intel CORE I9-13900K S1700 може значно прискорити час транзакцій і зробити покупців більше щасливими, а співробітників більше продуктивними.

Повне шифрування диска зашифровує диск повністю за винятком MBR. На додачу до Microsoft BitLocker, є ряд інших додатків шифрування диска, які можуть використовувати Intel CORE I9-13900K S1700, наприклад PGPdisk. Проблема з повним шифруванням диска полягає в тому, що воно може викликати зниження продуктивності, у результаті чого користувачі можуть відмовлятися від використання даного методу шифрування. З Intel CORE I9-13900K S1700 цей вплив на продуктивність практично зникає, і користувачі більш охоче будуть включати повне шифрування диска й використовувати його переваги.

Поліпшення продуктивності

Так які поліпшення продуктивності ми насправді побачимо з Intel CORE I9-13900K S1700? Поки що важко сказати точно, що дана технологія може нам запропонувати, оскільки вона досить нова. Але компанія Intel провела ряд власних випробувань, результати яких радують:

– При роботі з банківськими інтернет послугами на Microsoft IIS/PHP співробітники компанії виявили, що, порівнюючи дві системи на базі Nehalem,

одна із шифруванням і одна без, був приріст в 23% користувачів, яких можна підтримувати на цій системі. Коли система Nehalem із шифруванням рівнялася з non-Nehalem системою, поліпшення в кількості підтримуваних користувачів склалася 4.5 рази. Це разючі результати!

– У тесті шифрування / розшифровки бази даних Oracle 11g співробітники компанії виявили, що при порівнянні двох Nehalem систем, одна із включеним шифруванням, інша – ні, система із включеним шифруванням показала 89% зниження часу на розшифровку 5.1 мільйонів рядків зашифрованої таблиці. Також спостерігалося 87% зниження часу на зашифровку таблиць типу OLTP і повторну вставку й видалення одного мільйона рядків.

– Повне шифрування диска може займати масу часу для початкового шифрування диска. Компанія Intel виявила, що при шифруванні Intel 32 ГБ SDD диска в перший раз за допомогою шифрування кінцевої точки McAfee для ПК спостерігалося зниження часу першого заповнення на 42%. Це просто разюча різниця, що ви виразно відчуєте, якщо вам до цього доводило чекати закінчення процесу повного шифрування диска в перший раз.

3.2 Розробка структурної схеми

Для реалізації системи використовуються команди процесора Intel CORE I9-13900K S1700. Процесори насправді знаменують собою зміну поколінь, оскільки при цьому не тільки відбувається перехід на наступний техпроцес (32 нм у порівнянні з 45 нм), але перед нами й перше покоління CPU з підтримкою декількох інструкцій, що прискорюють шифрування. Intel згадує добавку як AES New Instructions. Вони складаються із:

– інструкцій для шифрування AES (AESENC, AESENCLAST);

– інструкцій для розшифровки (AESDEC, AESDECLAST);

– інструкції для роботи із ключем AES (AESIMC, AESKEYGENASSIST).

Як і раніше, інструкції відносяться до SIMD, тобто до типу "одна інструкція багато даних" (Single Instruction Multiple Data). Підтримуються всі три ключі AES (128, 192 і 256 біт з 10, 12 і 14 проходами підстановки й перестановки). Оскільки всі інструкції AES мають фіксовану затримку, що не залежить від даних, тобто час фіксований й доступ до пам'яті не потрібно. Крім того, модель програмування така ж, як і у випадку інших інструкцій SSE з первісного стандарту SSE4. Таким чином, всі операційні системи, які підтримують роботу з SSE, зможуть використовувати й інструкції AES New Instructions.

На рисунку 3.1 зображена структурна схема системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Виходячи зі структурної схеми системи зображеної на рисунку 3.1, система кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700, працює наступним чином.

Спершу при вході в систему, користувач звертається до блоку розмежування доступу системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700. Блок розмежування доступу отримує пароль користувача, та звертається до менеджеру паролів, де отримує сеансовий пароль перевірки правильності паролю користувача, та правильності прав доступу користувача, які зберігаються у відповідних зашифрованих базах даних. Розмежування цих баз зроблено з метою підвищення стійкості системи захищеного документообігу. Після підтвердження прав доступу, та правильності введеного паролю, користувачеві видається сеансовий ключ AES з використанням спеціальних інструкцій Intel CORE I9-13900K S1700 для роботи з інформацією.

У блоці шифрування згідно прав доступу, з отриманого ключа AES з використанням спеціальних інструкцій Intel CORE I9-13900K S1700 відбувається його розширення, та обирається ключ ітерації, за допомогою яких й відбувається шифрування інформації алгоритмом AES з використанням спеціальних інструкцій Intel CORE I9-13900K S1700.

Процедура дешифрування відбувається аналогічним чином.

Забезпечення безпеки інформації при зберіганні й обробці більших інформаційних масивів – одна із самих актуальних проблем сучасних інформаційних технологій. Інтенсивний розвиток методів розподіленої обробки даних і різке збільшення обсягів інформації, що накопичується в комп'ютерних системах, привели останнім часом до кардинальної зміни методів довгострокового зберігання даних. Традиційні підходи до організації зберігання великих інформаційних масивів перестали задовольняти зрослим вимогам до ємності носіїв і швидкості доступу до даних. Всі частіше споживач довіряє зберігання своєї власної інформації зовнішнім центрам або мережам зберігання даних (так званий аутсорсинг). Одна з основних сфер застосування мережного зберігання даних – формування банків даних електронних документів, а також електронних архівів і бібліотек. Такі сховища даних можуть бути як публічними, так і обмеженого доступу, залежно від характеру документів, що накопичуються в них.

Для шифрування великих масивів даних, що поміщаються в зовнішні стосовно власника інформації сховища, ефективні лише симетричні схеми шифрування. Можливості їхнього практичного застосування, мабуть, визначаються можливостями організації схеми керування секретними ключами, для яких необхідно забезпечити виконання двох почасти суперечливих вимог: забезпечення високої схоронності ключів (зокрема, за рахунок резервування) і обмеження середовища їхнього поширення тільки тими пристроями, яким довіряє власник інформації. У зв'язку із цим у деяких випадках більше раціональним виглядає застосування схем відкритого шифрування, що дає можливість невизначеному колу осіб поміщати свої дані в сховище, але доступ до них залишати лише для власників секретного ключа. Така схема може бути корисна, наприклад, для систем електронної пошти або систем планування потоків завдань (workflows), де циркулюють переважно повідомлення невеликої довжини. Для таких схем тим більше необхідні механізми пошуку за шифрованим даними, що операції розшифрування в асиметричних криптосхемах, як відомо, виконуються на кілька порядків повільніше в порівнянні із симетричними.

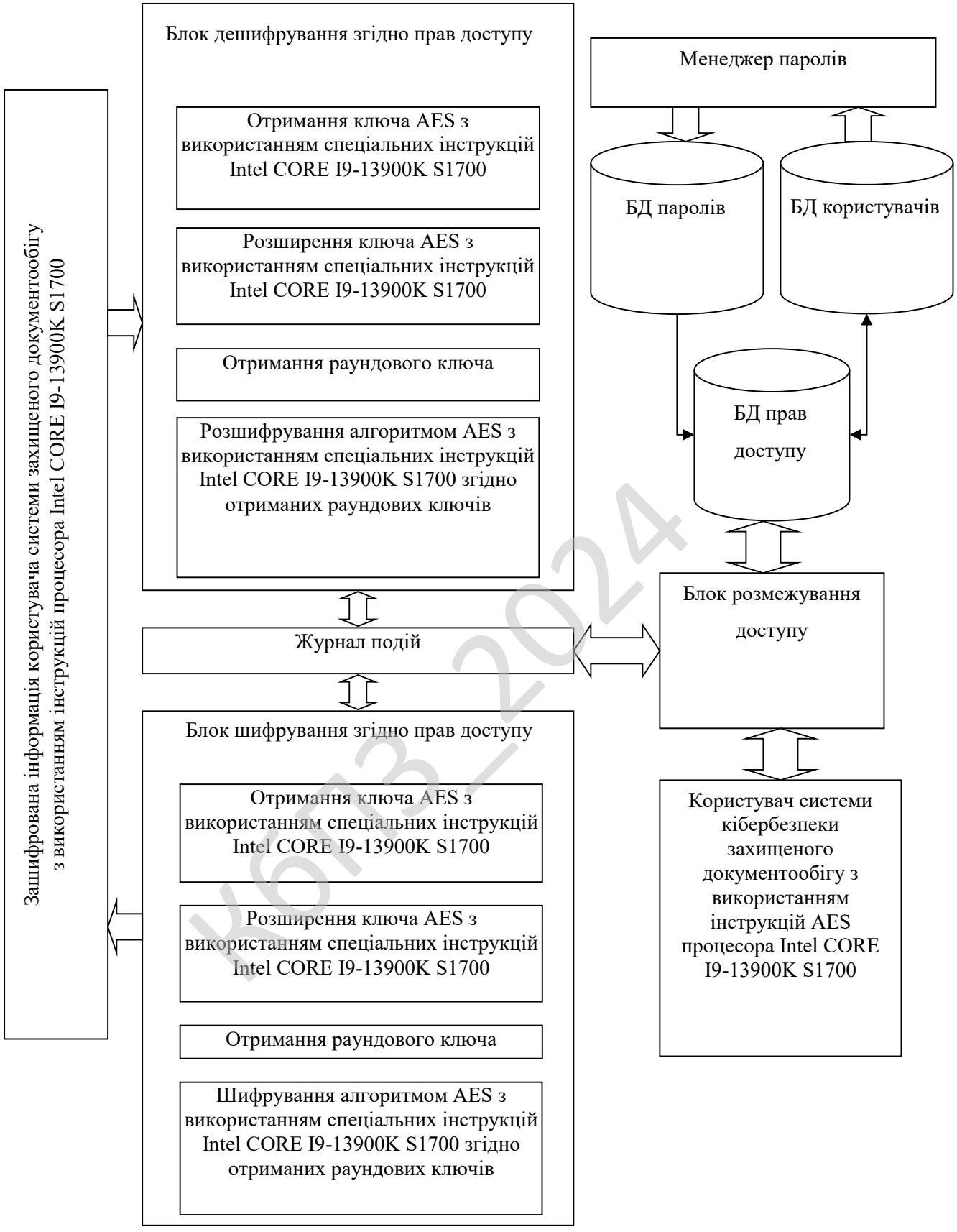

Рисунок 3.1 – Структурна схема системи

Інша проблема, пов'язана із забезпеченням конфіденційності пошуку в масивах даних, пов'язана з бажанням унеможливити одержання адміністратором СУБД і сторонніми особами відомостей про те, до яких саме записів (або фрагментів) бази даних здійснювався доступ при кожному конкретному запиті. У закордонній літературі це завдання зветься “Private Information Retrieval» (PIR). Вона особливо актуальна, наприклад, при обробці й зберіганні електронних документів, що містять відомості приватного характеру: фінансові, юридичні, майнові, медичні й інші.

Якщо навіть самі поля бази даних зашифровані, характер і частота запитів до них уже можуть дати зловмисникові деяку непряму інформацію, розголошення якої небажано для власника. Ці завдання до визначеної міри аналогічні виникаючої в телекомунікаційних системах завданню маскування інтенсивності трафіку між вузлами, що, як відомо, вирішується шляхом суцільного заповнення каналу псевдовипадковими послідовностями.

Нерідко перед приміщенням документів у мережні сховища вони піддаються стиску або іншим спеціальним видам кодування. У зв'язку із цим загострюється необхідність забезпечення керованості, надійності й безпеці зберігання й доступу до електронних документів, а також процедур їхньої передачі між прикладними програмами й пристроями зберігання.

Якщо навіть дані зберігаються локально, виникає інша проблема: адміністратори, що управляють СУБД і персонал так чи інакше звичайно мають права доступу до всієї збереженої інформації. Для захисту від їхніх несанкціонованих дій у деяких випадках доцільне застосування апаратно-програмних засобів шифрування даних перед записом їх на засоби зберігання.

Часто шифрувальні модулі вбудовуються, наприклад, у засоби резервного копіювання даних. Однак при зберіганні шифрованих масивів утруднений пошук окремих файлів і оперативний доступ до елементів масиву, необхідним для роботи прикладних програм. Так як масив зберігається в зашифрованому виді, і серверу, на якому він зберігається (або СУБД), не можуть бути довірені ключі шифрування, користувач (або прикладна програма від його ім'я) змушений

завантажувати копії всіх файлів масиву, розшифровувати їх і потім виконувати пошук на локальній машині. Очевидно, що такий спосіб пошуку дуже неекономічний. У зв'язку із цим вимальовується проблема забезпечення можливості пошуку даних по шифрованим і (або) стислим даним, що може бути конкретизована залежно від застосовуваної в системі моделі шифрування даних.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми.

– Блоки шифрування та дешифрування інформації згідно алгоритму AES з використанням спеціальних інструкцій Intel CORE I9-13900K S1700.

– Універсальне робоче місце.

– Мобільне робоче місце.

– Діловодство.

– Факс.

– Засідання.

– Договори.

– Нормативні й розпорядницькі документи.

– Доручення.

– Клієнти й Контакти.

– Звернення громадян.

– Допомога.

– Планування.

– Керування Персоналом.

– Корпоративний тренінг.

– WorkFlow.

– Захист.

Розглянемо ці блоки більш детально.

Головне вікно програми

Головне вікно призначене для швидкого доступу до основних функцій програми й меню. Програма складається з головного вікна, розташованого у верхній частині екрана й набору незалежних дочірніх вікон.

Розташування й розміри вікон можна змінювати за допомогою миші. Також існує можливість закрити непотрібні дочірні вікна (знову відобразити їх можна шляхом вибору відповідних пунктів у меню натисканням на аналогічні кнопки в головному вікні програми). Всі зроблені зміни збережуться в наступному сеансі роботи. Призначення всіх кнопок у програмі пояснюється спливаючими підказками: підведіть покажчик миші до будь-якої кнопки й затримаєте його – з'явиться спливаюча підказка із призначенням кнопки. Головне меню надає доступ до основних списків і функцій системи.

Блоки шифрування та дешифрування інформації згідно алгоритму AES з використанням спеціальних інструкцій Intel CORE I9-13900K S1700

Призначені для шифрування та дешифрування інформації, до якої користувач має доступ, згідно прав доступу.

При реалізації шифрування та дешифрування виконуються такі основні операції:

– Key Expansion – процедура використовувана для генерації Round Keys з Cipher Key.

– Cipher Key – секретний, криптографічний ключ, що використовується Key Expansion процедурою, щоб зробити набір ключів для раундів (Round Keys); може бути представлений як прямокутний масив байтів, що має чотири рядки й Nk колонок.

– Round Key – Round Keys виходять із Cipher Key використовуючи процедуру Key Expansion. Вони застосовуються до State при шифруванні й розшифруванні.

– State – проміжний результат шифрування, що може бути представлений як прямокутний масив байтів що має 4 рядки й Nb колонок.

– AddRoundKey() – трансформація при шифруванні й зворотному шифруванні, при якій Round Key XOR’иться с State. Довжина RoundKey дорівнює розміру State (тобто, якщо Nb = 4, то довжина RoundKey дорівнює 128 біт або 16 байт).

– SubBytes() – трансформації при шифруванні які обробляють State використовуючи нелінійну таблицю заміщення байтів (S-box), застосовуючи її незалежно до кожного байта State.

– ShiftRows() – трансформації при шифруванні, які обробляють State, циклічно зміщаючи останні три рядки State на різні величини.

– MixColumns() – трансформація при шифруванні яка бере всі стовпці State і змішує їх дані (незалежно друг від друга), щоб одержати нові стовпці.

– InvShiftRows() – трансформація при розшифруванні яка є зворотною стосовно ShiftRows().

– InvSubBytes() – трансформація при розшифруванні яка є зворотною стосовно SubBytes().

– InvMixColumns() – трансформація при розшифруванні яка є зворотною стосовно MixColumns().

– RotWord() – функція, що використовується в процедурі Key Expansion, що бере 4-х байтне слово й робить над ним циклічну перестановку.

– SubWord() – функція, використовувана в процедурі Key Expansion, що бере на вході 4-х байтне слово й застосовуючи S-box до кожного із чотирьох байтів видає вихідне слово.

– Block – послідовність біт, з яких складається input, output, State і Round Key. Також під Block можна розуміти послідовність байт.

– Ciphertext – вихідні дані алгоритму шифрування.

– S-box – нелінійна таблиця замін, що використовується в декількох трансформаціях заміни байт і в процедурі Key Expansion для взаємнооднозначної заміни значення байта.

– Nb – число стовпців(32-ух бітних слів), що становлять State. Для, AES Nb = 4.

– Nk – число 32-ух бітних слів, що становлять шифроключ. Для AES, Nk = 4,6, або 8.

– Nr – число раундів, що є функцією Nk і Nb. Для AES, Nr = 10, 12, 14.

– Rcon[] – масив, що складається з бітів 32-х розрядного слова і є постійним для даного раунду.

Крім того функціонально «Захищений електронний документообіг Intel CORE I9-13900K S1700» включає системи:

– що автоматизують ведення діловодства й керування документообігом («Захищений електронний документообіг Intel CORE I9-13900K S1700 – Діловодство», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Документи», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Інформаційно-довідкова система», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Факс», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Договори», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Звернення громадян», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Workflow»);

– що забезпечують інформаційну підтримку робочих процесів, властивих будь-який організації («Захищений електронний документообіг Intel CORE I9-13900K S1700 – Клієнти й контакти», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Планування», «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Засідання»);

– що автоматизують роботу кадрової служби («Захищений електронний документообіг Intel CORE I9-13900K S1700 – Керування персоналом»);

– що автоматизують взаємодію користувачів з відділом технічної підтримки (система «Захищений електронний документообіг Intel CORE I9-13900K S1700 – HelpDesk»);

– що забезпечують можливість навчання роботі в системі «Захищений електронний документообіг Intel CORE I9-13900K S1700» («Захищений електронний документообіг Intel CORE I9-13900K S1700 – Корпоративний тренінг»).

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Універсальне робоче місце

«Захищений електронний документообіг Intel CORE I9-13900K S1700 – Універсальне робоче місце» – це єдиний інтерфейс для всіх співробітників організації, що беруть участь в електронному документообігу: більш зручно ніж файлова система, і більш надійно ніж електронна пошта.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Мобільне робоче місце

«Захищений електронний документообіг Intel CORE I9-13900K S1700 – Мобільний портал» – робоче місце, що дозволяє здійснювати мобільний доступ до баз дані системи електронного документообігу «Захищений електронний документообіг Intel CORE I9-13900K S1700».

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Діловодство

Модуль «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Діловодство» призначений для автоматизації документообігу в територіально-розподілених організаціях, технологія діловодства яких припускає централізоване відстеження руху документів у реальному масштабі часу.

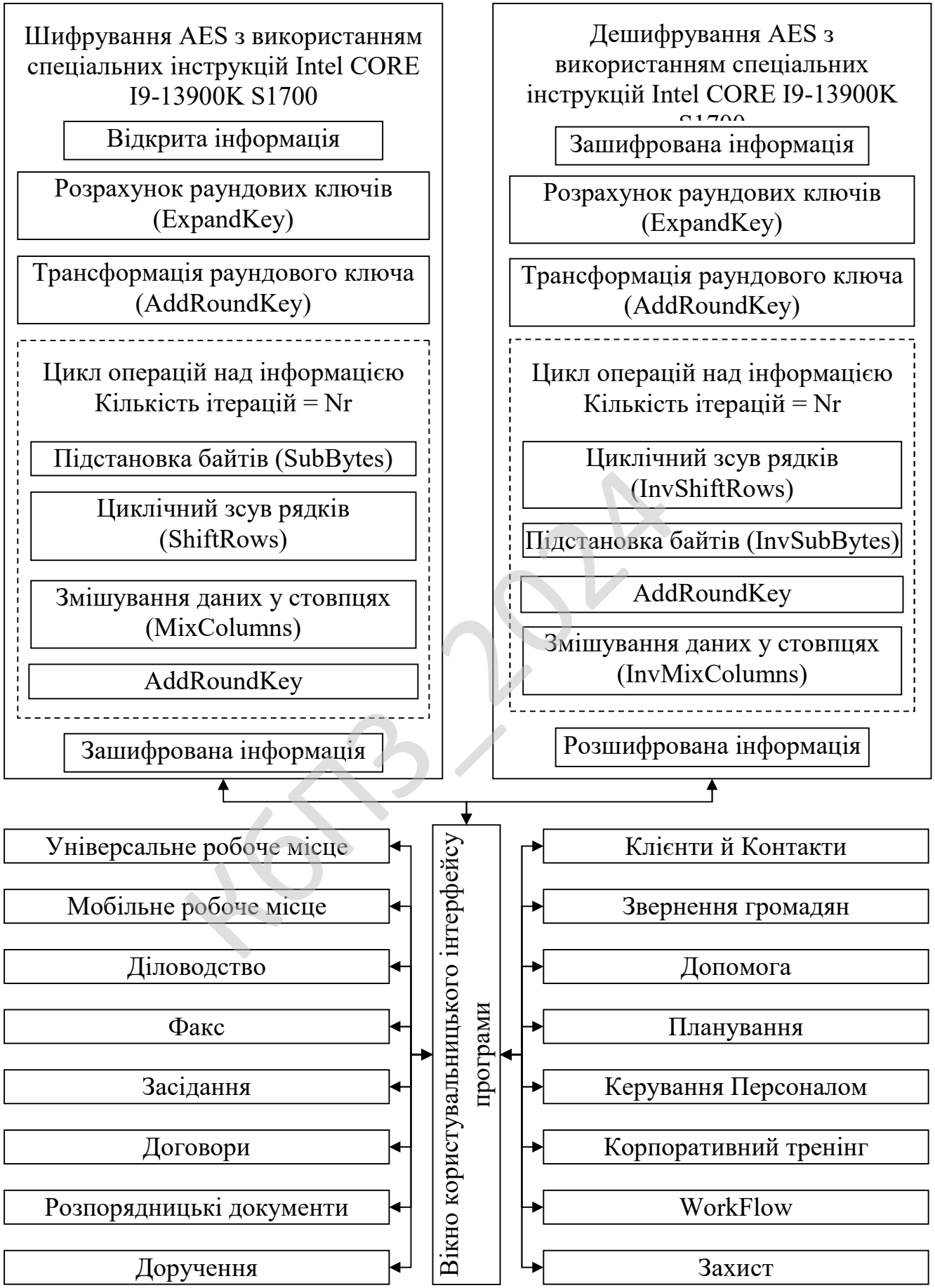

Рисунок 3.2 – Функціональна схема системи

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Факс

Система факс-серверної служби організації в складі системи корпоративного документообігу й діловодства «Захищений електронний документообіг Intel CORE I9-13900K S1700» істотно підвищує ефективність і зручність роботи з факсимільними повідомленнями.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Засідання

Підсистема призначена для автоматизації процесу документообігу, що супроводжує проведення засідань і нарад колегіальних органів керування організації.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Договори

Система призначена для ведення реєстру договорів і контролю виконання стосовних до них доручень.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Нормативні й розпорядницькі документи

Система «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Нормативно-розпорядницькі документи» призначена для публікації й зберігання офіційних діючих і застарілих нормативних, а також організаційно-розпорядницьких документів (уставів, положень, наказів, правил, інструкцій і т.д.), призначених для використання в інформаційно-довідкових цілях.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Доручення

Підсистема «Захищений електронний документообіг Intel CORE I9-13900K S1700 – Доручення» призначена для створення й контролю виконання усних доручень, не пов'язаних з документами.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Клієнти й Контакти

«Захищений електронний документообіг Intel CORE I9-13900K S1700 – Клієнти й контакти» дозволяє впорядкувати всю інформацію про клієнтів і дає можливість зберігати інформацію із всіх зовнішніх організацій, з якими коли-або контактувала компанія.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Звернення громадян

Система орієнтована на підприємства й організації, що ведуть діловодство на підставі звернень громадян і дозволяє скоротити часові витрати на реєстрацію, обробку й контроль виконання заявок фізичних осіб.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Допомога

Система призначена для автоматизації процесу рішення технічних проблем, які виникають у користувачів компанії при експлуатації технічного й програмного забезпечення, для ведення архіву усунутих проблем з метою їхнього подальшого аналізу, для узагальнення й використання результатів при виникненні аналогічних ситуацій, для обліку завантаження співробітників по проектах, для короткострокового планування роботи на майбутній період.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Планування

Система призначена для планування й координації робіт, для здійснення контролю за виконанням поставлених планів, для нагромадження інформації про успішні й провалені проекти з метою їхній наступного аналізу. Система дозволяє підвищити ефективність управлінської діяльності компанії.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Керування Персоналом

Система призначена для автоматизації виробничих процесів, пов'язаних з кадровим обліком і кадровим документообігом.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Корпоративний тренінг

«Захищений електронний документообіг Intel CORE I9-13900K S1700 – Корпоративний тренінг» – це програмно-методичний комплекс навчання й атестації користувачів і інструкторів системи «Захищений електронний документообіг Intel CORE I9-13900K S1700».

Захищений електронний документообіг Intel CORE I9-13900K S1700 – WorkFlow

Термін workflow дослівно означає «потік робіт». Однак технологія workflow розглядається набагато ширше – це автоматизація робітників бізнес-процесів. Бізнес-процес, по суті справи, поєднує в собі все: потік робіт і функції, людей і встаткування, що реалізує ці функції, а також правила, керуючі послідовністю цих функцій.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Захист

Програмний продукт Захист призначений для підключення зовнішніх засобів криптозахисту інформації.

Захищений електронний документообіг Intel CORE I9-13900K S1700 – Центр Звітів

Система «Центр звітів» призначена для створення звітів за даними, що зберігається в базах даних. За допомогою цієї системи можна створювати довільно оформлені звіти, що дозволяє швидко одержати інформацію в зручній для користувача формі. Можливість створення власних звітів дозволяє враховувати особливості роботи будь-якої організації.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

Після початку роботи розробленого ПЗ ми потрапляємо до інтерфейсу ПЗ звідки можемо потрапити до бібліотека взаємодії з інструкціями процесора Intel CORE I9-13900K S1700 та через моніторинг дискової підсистеми та введення ключа шифрування можемо проводити шифрування та дешифрування. Якщо проходе шифрування, проводиться збереження зашифрованого файлу, виведення часу шифрування, виведення повідомлення про завершення шифрування, відкриття файлу для шифрування та виведення інформації про файл до шифрування. Якщо проходе дешифрування проводиться збереження дешифрованого файлу, виведення часу дешифрування, виведення повідомлення про завершення дешифрування, відкриття файлу для дешифрування та виведення інформації про файл до дешифрування.

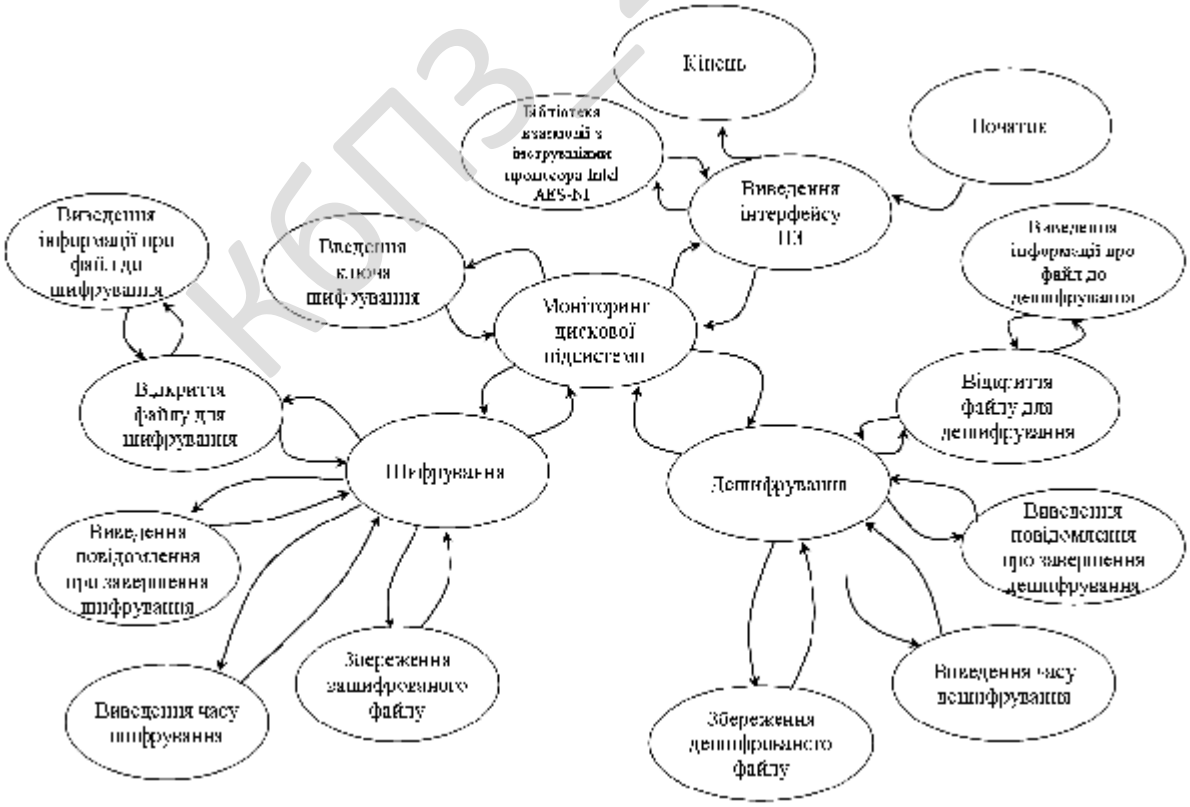

Рисунок 3.3 – Діаграма взаємодії процесів

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ 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:

– Ініціалізація та виведення інтерфейсу ПЗ.

– Запит шифрування даних?

– Обрання файлу або групи файлів.

– Аналіз та виведення назви та розміру кінцевого файлу.

– Введення користувачем ключа шифрування.

– Підпрограма роботи з інструкціями Intel CORE I9-13900K S1700.

– Виведення повідомлення результату шифрування.

– Виведення кінцевих параметрів даних.

– Обрання шляху та збереження даних.

– Запит дешифрувати дані?

– Обрання файлу дешифрування**.**

– Аналіз та виведення інформації контейнера.

– Введення ключа шифрування.

– Підпрограма роботи з інструкціями Intel CORE I9-13900K S1700.

– Запит дешифрування успішно?

– Виведення повідомлення результату шифрування.

– Виведення параметрів проведеного дешифрування.

– Збереження дешифрованого файлу.

– Запит WM_CLOSE?

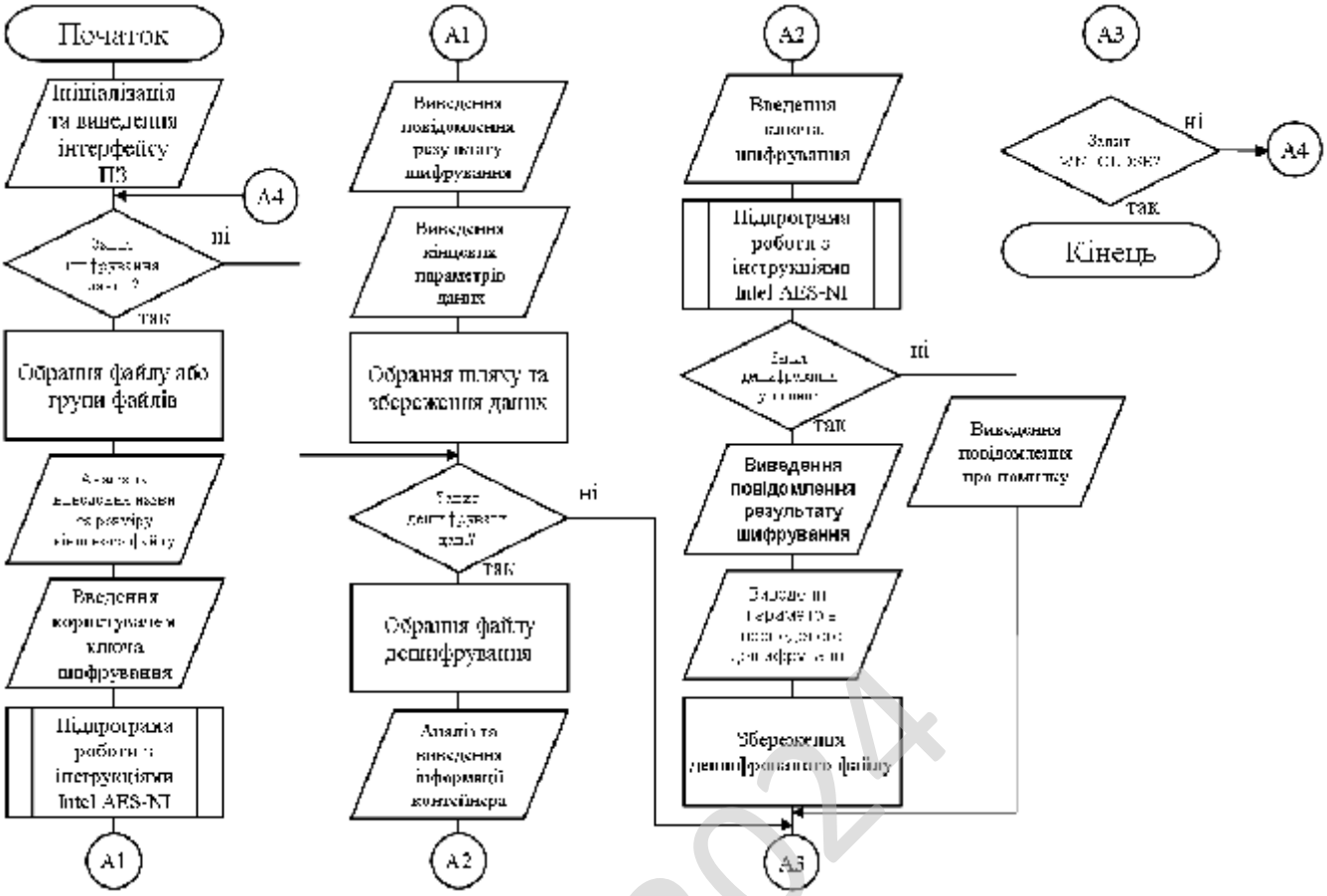

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 наведено блок-схему підпрограми роботи з інструкціями Intel CORE I9-13900K S1700. Її робота складається з виконання наступних кроків:

– Запит шифрувати дані?

– Сканування та розбиття даних на блоки.

– Формування сеансового ключа з ключа шифрування.

– Перемішування байтів відповідно до S-box.

– Формування масиву байтів.

– Циклічний зсув рядків та перемішування колонок.

– Перетворення над даними з використанням сеансового ключа.

– Формування зашифрованих даних.

– Запит дешифрувати дані?

– Розбиття даних для дешифрування на блоки.

– Формування сеансового ключа з ключа шифрування.

– Зворотний циклічний зсув рядків.

– Зворотне перемішування байтів відповідно до S-box.

– Перетворення над даними з використанням сеансового ключа.

– Зворотне перемішування колонок.

– Формування дешифрованих даних.

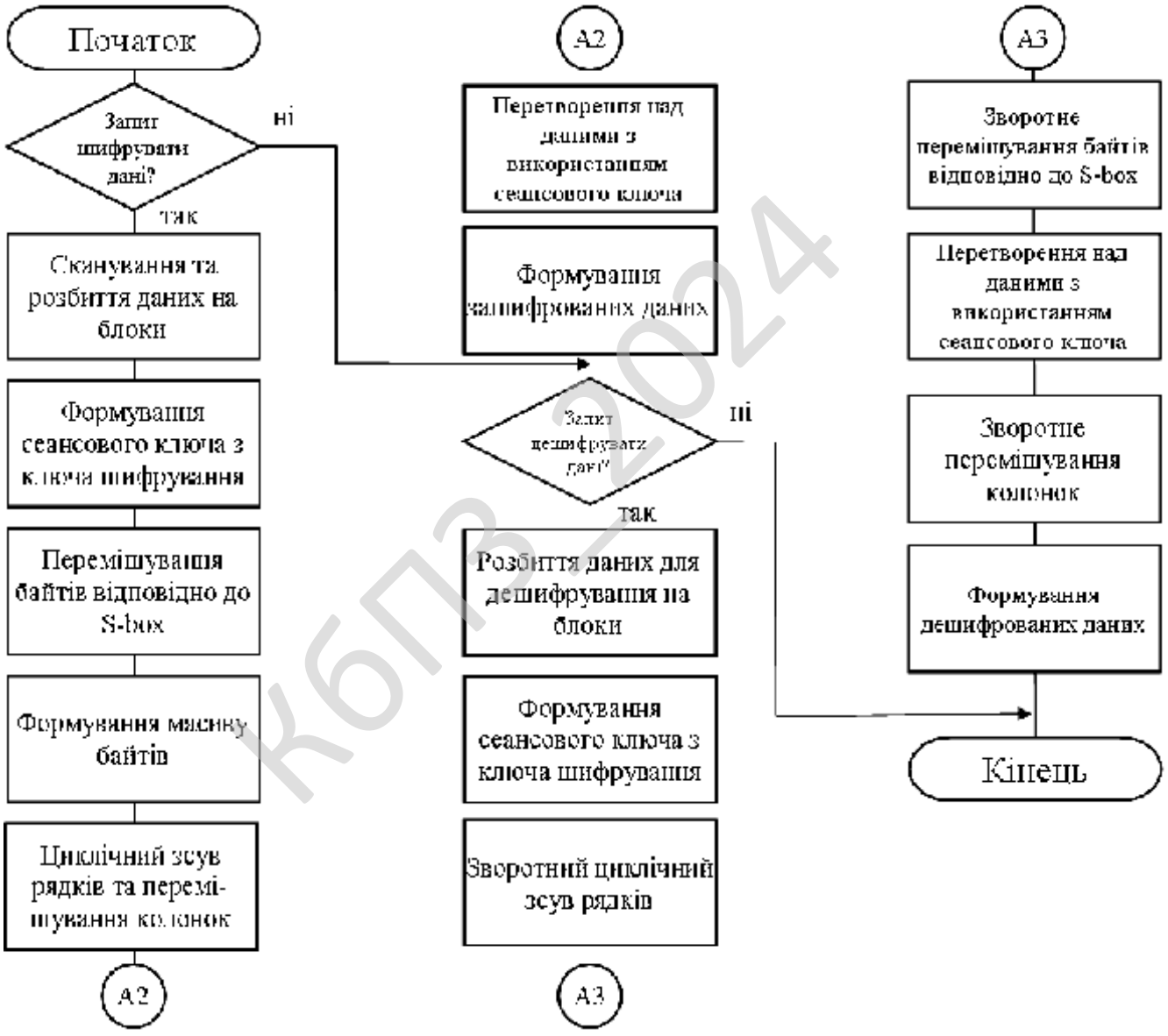

Рисунок 4.2 – Блок-схема підпрограми роботи з інструкціями Intel CORE I9-13900K S1700

Розглянемо реалізовані алгоритми та методи при вирішені питань які виникли при роботі над дипломом.

AES розшифровується як "Advanced Encryption Standard" – це найбільш популярний стандарт симетричного шифрування у світі IT. Стандарт працює із блоками розміром 128 біт і підтримує 128-, 192– або 256-бітні ключі (AES-128, AES-192 і AES-256). Багато утиліт шифрування, та ж TrueCrypt, підтримали алгоритм AES на самому початку його існування. Але найбільший фактор успіху AES, звичайно, полягає в його прийнятті урядом США в 2002 році, при цьому в 2003 році він був прийнятий як стандарт для захисту секретних даних. Intel CORE I9-13900K S1700 вперше було реалізовано по технології Westmere.

Westmere це 32нм техпроцес – площа кристалів менша, що дозволило збільшити кількість ядер. Перший представник нової мікроархітектури – Clarkdale, з двома ядрами та інтегрованим графічним ядром, виробленим за 45-нм техпроцесом, що дозволило позбавитися від інтегрованої графіки в системній логіці. Процесори на основі дизайну Clarkdale створені під LGA1156, але для реалізації інтегрованої графіки потрібно спеціальні набори системної логіки, в них входять Intel H55, Intel H57 і Intel Q57. Це рішення замінить собою процесори на основі Wolfdale (Core 2 Duo). Продукти на основі дизайну ядер Clarkdale надійшли у відкритий продаж 7 січня 2010 року.

Потім в серійне виробництво увійшов флагманський дизайн ядер даної архітектури – Gulftown, з шістьма ядрами, дванадцятьма потоками, 12 Мб загального кеша третього рівня, системною шиною QuickPath Interconnect, але попри це, його енергоспоживання не перевищує 130 Вт. Він вимагає роз'єму LGA1366 і набір системної логіки Intel X58 Express. Фактично цей дизайн являє собою півтора чипа з дизайном Bloomfield (Core i7) на одній підкладці, виробленої з дотриманням норм 32-нм техпроцесу. Цей дизайн ядер є першим, який переступив за психологічну позначку – один мільярд транзисторів. Він має 1,17 млрд транзисторів, однак за рахунок 32-нм техпроцесу його площа залишилася в розумних межах – 245мм².

На основі цього дизайну вийшов процесор Core i7 980X Extreme Edition з частотою 3333 МГц і Core i7 970 з частотою 3200 МГц (3 квартал 2010 року), а також Core i7 990X Extreme Edition з частотою 3466 МГц (1 квартал 2011 року). Процесори на основі дизайну ядер Gulftown надійшли у відкритий продаж 16 березня 2010 року.

Шифрування даних за допомогою AES. Шифрування AES базується на системі підстановок з перестановкою, тобто над даними проводиться серія математичних операцій, щоб створити значно модифікований масив даних (зашифрований). Як вихідна інформація виступає текст, а ключ відповідає за виконання математичних операцій. Операції можуть бути як зовсім простими, наприклад, зрушення бітів або XOR, так і більше складними. Один прохід можна легко розшифрувати, тому всі сучасні алгоритми шифрування побудовані на декількох проходах. У випадку AES це 10, 12 або 14 проходів для AES-128, AES-192 або AES-256. До речі, ключі AES проходять таку ж процедуру, що й користувальницькі дані, тобто вони являють собою що змінюється раундовий ключ.

Процес працює з масивами 4x4 з одиночних байтів, також називаних боксами: S-box використовуються для підстановок, P-box – для перестановок. Підстановки й перестановки виконуються на різних етапах: підстановки працюють усередині так званих боксів, а перестановки міняють інформацію між боксами. S-box працює по складному принципі, тобто навіть якщо єдиний вхідний біт буде мінятися, то це вплине на декілька вихідних бітів, тобто властивості кожного вихідного біта залежать від кожного вхідного біта.

Використання декількох проходів забезпечує гарний рівень шифрування, при цьому необхідно відповідати критеріям розсіювання (diffusion) і заплутування (confusion). Розсіювання виконується через каскадну комбінацію трансформацій S-box і P-box: при зміні тільки одного біта у вхідному тексті S-box буде модифікувати вихід декількох біт, а P-box буде псевдовипадково поширювати цей ефект по декількох S-box. Коли ми говоримо про те, що

мінімальна зміна на вході дає максимальна зміна на виході, ми говоримо про ефект сніжної грудки.

Надійність шифрування AES. Останнім часом іде чимало обговорень так званих взламів, які обходять необхідність запуску розширеного пошуку методом грубої сили для знаходження правильного ключа розшифровки. Технології, такі як атаки XSL і атаки related-key обговорюються досить інтенсивно – але успіх невеликий. Єдиний працюючий спосіб злому шифрування AES полягає в так званій атаці побічного каналу (side-channel). Для її здійснення атака повинна відбуватися тільки на host-системі, на якій виконується шифрування AES, і при цьому вам необхідно знайти спосіб одержання інформації про синхронізацію кеша. У такому випадку можна відстежити число тактів комп'ютера до завершення процесу шифрування.

Звичайно, все це не так легко, оскільки вам потрібен доступ до комп'ютера, причому досить повний доступ для аналізу шифрування й права на виконання коду. Тепер вам напевно зрозуміло, чому "діри" у системі безпеки, які дозволяють зловмисникові одержати такі права, нехай навіть вони звучать зовсім абсурдно, необхідно закривати якнайшвидше. Але якщо ви одержите доступ до цільового комп'ютера, то добування ключа AES – справа часу, тобто вже не трудомістке завдання для суперкомп'ютерів, що вимагає величезних обчислювальних ресурсів.

AES усередині Intel. На даний момент інтегровані в CPU інструкції AES починають мати сенс – незалежно від можливих переваг по продуктивності. З погляду безпеки процесор може обробляти інструкції AES в інкапсульованому виді, тобто йому не потрібні які-небудь таблиці перетворення, необхідні для атаки методом побічного каналу.

Процесори насправді знаменують собою зміну поколінь, оскільки при цьому не тільки відбувається перехід на наступний техпроцес (32 нм у порівнянні з 45 нм), але перед нами й перше покоління CPU з підтримкою декількох інструкцій, що прискорюють шифрування. Intel згадує добавку як AES New

Instructions. Вони складаються із чотирьох інструкцій для шифрування AES (AESENC, AESENCLAST) і розшифровки (AESDEC, AESDECLAST) плюс ще дві інструкції для роботи із ключем AES (AESIMC, AESKEYGENASSIST). Як і раніше, інструкції відносяться до SIMD, тобто до типу "одна інструкція багато даних" (Single Instruction Multiple Data). Підтримуються всі три ключі AES (128, 192 і 256 бітів з 10, 12 і 14 проходами підстановки й перестановки).

Оскільки всі інструкції AES мають фіксовану затримку, що не залежить від даних, тобто час фіксоване й доступ до пам'яті не потрібно. Крім того, модель програмування така ж, як і у випадку інших інструкцій SSE з первісного стандарту SSE4. Таким чином, всі операційні системи, які підтримують роботу з SSE, зможуть використовувати й інструкції AES New Instructions.

Будьте обережні при виборі процесора із прискоренням AES, оскільки сьогодні лише деякі моделі підтримують нові інструкції. 32-нм процесори Core i3 на Clarkdale не підтримують інструкції, а дводерна лінійка Core i 5-600 – підтримує. У мобільних процесорів ситуація ледве більше складна: якщо мобільні процесори Core i3 теж не підтримують прискорення AES, те процесори лінійки Core i 5-500 уже підтримують. Однак є одна модель Core i 5-400, позбавлена такої підтримки. Усе було б набагато простіше, якби Intel додала підтримку нових інструкцій в усі моделі.

Наведемо код, який призначений для шифрування.

```
function EncryptData(Data: string; AKey: AnsiString; AIv: AnsiString): string;
// функція шифрування вхідних даних
        var
          cipher: TDCP_rijndael;
          key, iv, src, dest, b64: TBytes;
          index, slen, bsize, pad: integer;
        begin
         //key := Base64DecodeBytes(TEncoding.UTF8.GetBytes(AKey));
          //iv := Base64DecodeBytes(TEncoding.UTF8.GetBytes(AIv));
          key := TEncoding.ASCII.GetBytes(AKey);
          iv := TEncoding.ASCII.GetBytes(AIv);
          src := TEncoding.UTF8.GetBytes(Data);
          cipher := TDCP_rijndael.Create(nil);
```

```
        try
          cipher.CipherMode := cmCBC;
          slen := Length(src);
          bsize := (cipher.BlockSize div 8);
          pad := bsize - (slen mod bsize);
          Inc(slen, pad);
          SetLength(src, slen);
          for index := pad downto 1 do
          begin
            src[slen - index] := pad;
          end;
          SetLength(dest, slen);
          cipher.Init(key[0], 256, @iv[0]);
      // DCP uses key size in BITS not BYTES
          cipher.Encrypt(src[0], dest[0], slen);
          b64 := Base64EncodeBytes(dest);
          result := TEncoding.Default.GetString(b64);
        finally
          cipher.Free;
        end;
      end;
```

Наведемо код, який призначений для дешифрування.

```
      function DecryptData(Data: string; AKey: AnsiString; AIv: AnsiString):
string;
      var
        key, iv, src, dest: TBytes;
        cipher: TDCP_rijndael;
        slen, pad: integer;
      begin
        //key := Base64DecodeBytes(TEncoding.UTF8.GetBytes(AKey));
        //iv := Base64DecodeBytes(TEncoding.UTF8.GetBytes(AIv));
          slen := Length(src);
          bsize := (cipher.BlockSize div 8);
          pad := bsize - (slen mod bsize);
          Inc(slen, pad);
          SetLength(src, slen);
        key := TEncoding.ASCII.GetBytes(AKey);
        iv := TEncoding.ASCII.GetBytes(AIv);
        src := Base64DecodeBytes(TEncoding.UTF8.GetBytes(Data));
        cipher := TDCP_rijndael.Create(nil);
        try
```

```
            cipher.CipherMode := cmCBC;
            slen := Length(src);
            SetLength(dest, slen);
            cipher.Init(key[0], 256, @iv[0]);    cipher.Decrypt(src[0], dest[0],
slen);
            pad := dest[slen - 1];
            SetLength(dest, slen - pad);
            result := TEncoding.Default.GetString(dest);
          finally
            cipher.Free;
          end;
        end;
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

$$Tag=H_{K1}(M)\oplus F_{K2}(Nonce)$$

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

Вибір блокового шифру

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16(тому що AES працює з 16-байтними блоками).

KDF – функція створення ключа

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

PDF: функція створення псевдовипадкового числа

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

Генерація UMAC-тегів

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

UMAC-32 UMAC-64 UMAC-96 UMAC-128

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 (K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 (K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 (K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

Універсальна функція гешування(UHASH)

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і,

нарешті, до результату застосовується L3-HASH . Якщо при цьому довжина вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

Універсальна функція

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

L1-hash – перший етап

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

L2-hash – другий етап

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

L3-hash – третій етап

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Файл; Шаблон; Налаштування; Довідка.

– Блок виведення даних: Шифрування/Дешифрування.

– Поточні дії програми: журнал роботи ПЗ.

– Функції ПЗ: Шифрувати / Дешифрувати.

– Функції ПЗ: Перевірити файл.

– Функції ПЗ: Дані файлу.

– Функції ПЗ: Обрати ключі шифрування.

– Функції ПЗ: Налаштування.

– Функції ПЗ: Авторське право.

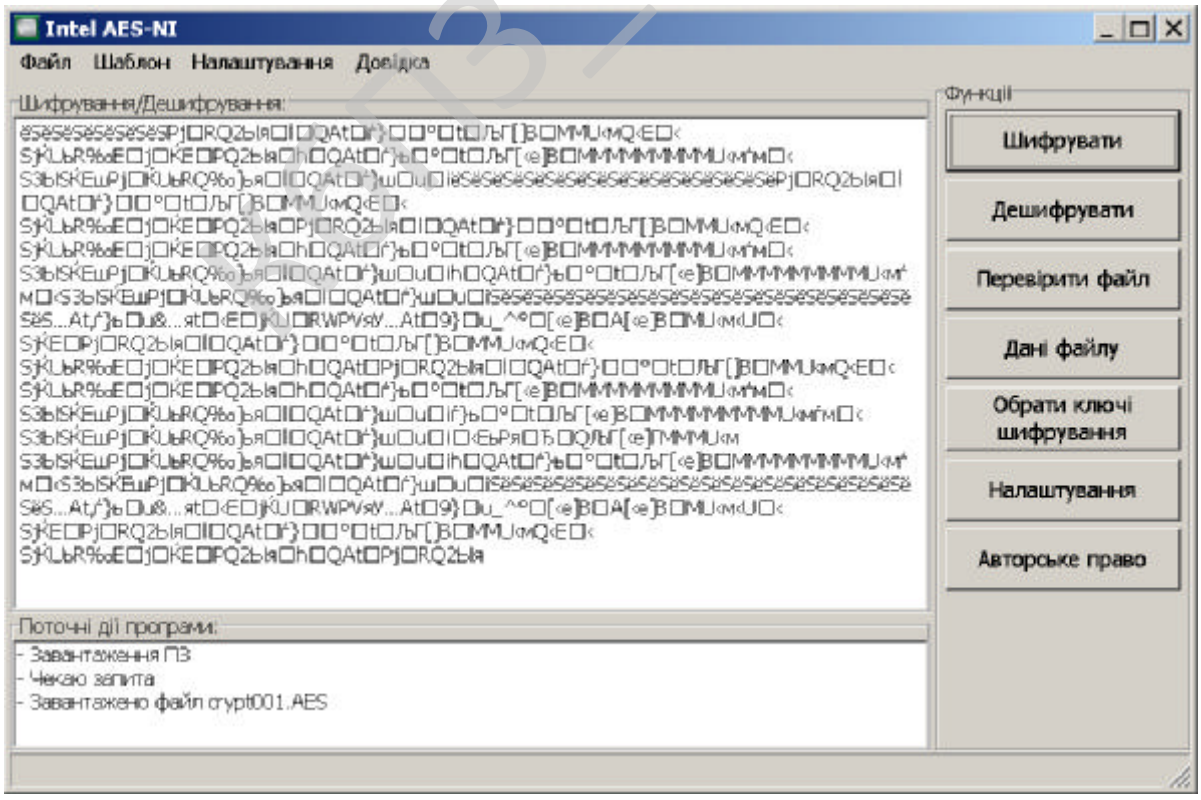

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права. Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy).

ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій. Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

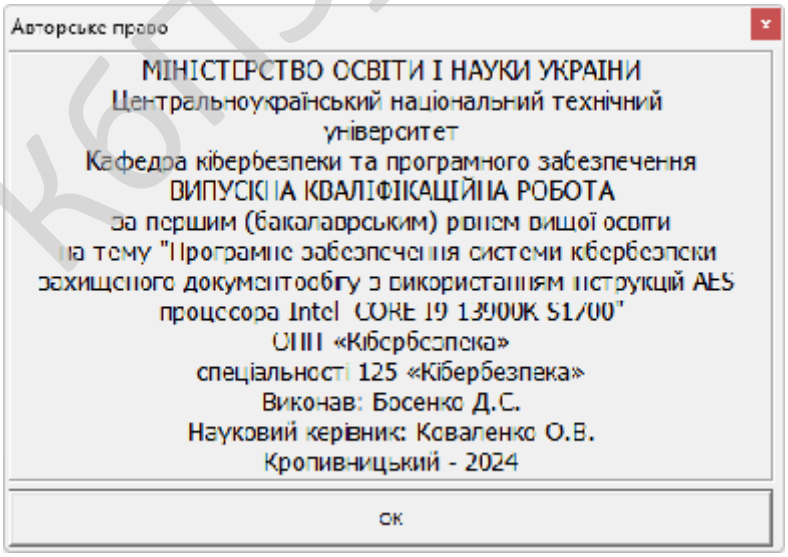

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

– Досліджена система захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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 RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700. 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 UMAC.

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. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

2. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

3. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

4. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

5. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

6. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

7. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

8. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

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

10. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

11. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

12. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

13. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

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

15. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

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

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

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

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

21. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

22. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

23. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-

quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

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

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

29. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

30. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

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

32. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

33. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

34. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

35. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

36. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

37. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

38. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

39. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

40. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

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

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

43. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

44. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

45. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *ІІ Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

46. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

47. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

48. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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. № 136-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700.

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 захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700;

– 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, 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 RAD Studio Delphi 10.4.

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 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 13.06.2024 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

_____ Коваленко О.В.

Програмне забезпечення системи кібербезпеки захищеного документообігу з використанням інструкцій AES процесора Intel CORE I9-13900K S1700

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 51

Літера: РП

Кропивницький – 2024 року

Файл Main.pas – основна програма

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, El_Intel AES_NI, Math, Buttons;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    OpenDialog1: TOpenDialog;
    Button1: TButton;
    Button2: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Edit2: TEdit;
    Label_Time: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label_Status: TLabel;
    Memo_PlainText: TMemo;
    Memo_CyperText: TMemo;
    Label11: TLabel;
    Label12: TLabel;
    Memo_UncipherText: TMemo;
    Label13: TLabel;
    BitBtn_Encrypt: TBitBtn;
    BitBtn_Decypt: TBitBtn;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure BitBtn_EncryptClick(Sender: TObject);
    procedure BitBtn_DecyptClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;



var
  Form1: TForm1;
  EncryptedText: string;

function StringToHex(S: string): string; forward;
function HexToString(S: string): string; forward;

implementation

uses about;

{$R *.DFM}

function StringToHex(S: string): string;
var
      i: integer;
```

```
begin
  Result := '';

  // послідовно беремо одиничні символи та перетворюємо їх
  // до шістнадцяткових...
      for i := 1 to Length( S ) do
      Result := Result + IntToHex( Ord( S[i] ), 2 );
end;

function HexToString(S: string): string;
var
      i: integer;

begin
  Result := '';

  // послідовно беремо одиничні шістнадцяткові символи та перетворюємо їх
  // ASCII символів...
  for i := 1 to Length( S ) do
  begin
    // Туту обробляється тільки 2 шістнадцяткових блока...
      if ((i mod 2) = 1) then
            Result := Result + Chr( StrToInt( '0x' + Copy( S, i, 2 )));
  end;
end;


procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenDialog1.Execute then
    Edit1.Text := OpenDialog1.FileName;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  Source, Dest: TFileStream;
  SrcFile, DestFile: string;
  Start, Stop: cardinal;
  Size: integer;
  Key: T_Intel AES_NIKey128;
  SrcBuf, DstBuf: array [0..16383] of byte;
  SrcSize, DstSize: integer;
begin
  // Шифрування
  Label_Status.Caption := 'Шифруємо...';
  Refresh;
  Source := TFileStream.Create(Edit1.Text, fmOpenRead);
  try
    Label4.Caption := IntToStr(Source.Size div 1024) + ' KB';
    Refresh;
    DestFile := ExtractFilePath(Application.ExeName) + 'aestemp.enc';
    Dest := TFileStream.Create(DestFile, fmCreate);
    try
      Size := Source.Size;
      Dest.WriteBuffer(Size, SizeOf(Size));

      FillChar(Key, SizeOf(Key), 0);
      Move(PChar(Edit2.Text)^, Key, Min(SizeOf(Key), Length(Edit2.Text)));

      Start := GetTickCount;
      Encrypt_Intel AES_NIStreamECB(Source, 0, Key, Dest);
      Stop := GetTickCount;
      Label_Time.Caption := IntToStr(Stop - Start) + ' ms';
      Refresh;
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
```

```
  end;
  // Дешифрування
  Label_Status.Caption := 'Дешифруємо...';
  Refresh;
  Source := TFileStream.Create(DestFile, fmOpenRead);
  try
    Source.ReadBuffer(Size, SizeOf(Size));
    SrcFile := ExtractFilePath(Application.ExeName) + 'aestemp.dec';
    Dest := TFileStream.Create(SrcFile, fmCreate);
    try
      Start := GetTickCount;
      Decrypt_Intel AES_NIStreamECB(Source, Source.Size - Source.Position, Key,
Dest);
      Dest.Size := Size;
      Stop := GetTickCount;
      Label8.Caption := IntToStr(Stop - Start) + ' ms';
      Refresh;
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
  end;
  // Порівнюємо
  Label_Status.Caption := 'Порівнюємо...';
  Refresh;
  Source := TFileStream.Create(Edit1.Text, fmOpenRead);
  SrcSize := Source.Size;
  try
    Dest := TFileStream.Create(SrcFile, fmOpenRead);
    DstSize := Dest.Size;
    try
      repeat
        Source.ReadBuffer(SrcBuf, Min(SizeOf(SrcBuf), SrcSize));
        Dest.ReadBuffer(DstBuf, Min(SizeOf(DstBuf), DstSize));
        if not CompareMem(@SrcBuf, @DstBuf, Max(Min(SizeOf(DstBuf), DstSize),
Min(SizeOf(SrcBuf), SrcSize))) then
        begin
          ShowMessage('ПОМИЛКА: файл було змінено!!!');
          DeleteFile(SrcFile);
          DeleteFile(DestFile);
          exit;
        end;
        Dec(SrcSize, Min(SizeOf(SrcBuf), SrcSize));
        Dec(DstSize, Min(SizeOf(DstBuf), DstSize));
      until (SrcSize = 0) or (DstSize = 0);
      ShowMessage('Зміни не знайдено');
    finally
      Dest.Free;
    end;
  finally
    Source.Free;
  end;
  Label_Status.Caption := 'Видалено...';
  Refresh;
  Label_Status.Caption := '';
end;


(*****************************************************************************
   Використовуємо TStringStream... для шифрування
*****************************************************************************)
procedure TForm1.BitBtn_EncryptClick(Sender: TObject);
var
  Source: TStringStream;
  Dest: TStringStream;
  Start, Stop: cardinal;
  Size: integer;
  Key: T_Intel AES_NIKey128;
```

```
begin
  // Шифрування
  Label_Status.Caption := 'Шифруємо...';
  Refresh;
  Source := TStringStream.Create( Memo_PlainText.Text );
  Dest   := TStringStream.Create( '' );

  try
    // Зберігаємо дані до пам'яті...
    Size := Source.Size;
    Dest.WriteBuffer( Size, SizeOf(Size) );

    // Початковий ключ...
    FillChar( Key, SizeOf(Key), 0 );
    Move( PChar(Edit2.Text)^, Key, Min( SizeOf( Key ), Length( Edit2.Text )));

    // Починаємо шифрування...
    Починаємо := GetTickCount;
    Encrypt_Intel AES_NIStreamECB( Source, 0, Key, Dest );
    Stop := GetTickCount;
    Label_Time.Caption := IntToStr(Stop - Start) + ' ms';
    Refresh;

    // Видаємо зашифрований текст на екран...
        Memo_CyperText.Lines.BeginUpdate;
    Memo_CyperText.Text := StringToHex( Dest.DataString );
        Memo_CyperText.Lines.EndUpdate;

  finally
    Source.Free;
    Dest.Free;
  end;
end;

procedure TForm1.BitBtn_DecyptClick(Sender: TObject);
var
  Source: TStringStream;
  Dest: TStringStream;
  Start, Stop: cardinal;
  Size: integer;
  Key: T_Intel AES_NIKey128;
  EncryptedText: TStrings;
  S: string;

begin
  // перетворюємо шістнадцяткове число до рядка перед дешифруванням...
  Source := TStringStream.Create( HexToString( Memo_CyperText.Text ));
  Dest   := TStringStream.Create( '' );

  try
    // Починаємо дешифрування...
    Size := Source.Size;
    Починаємо := GetTickCount;
    Source.ReadBuffer(Size, SizeOf(Size));

    // Початковий ключ...
    FillChar(Key, SizeOf(Key), 0);
    Move(PChar(Edit2.Text)^, Key, Min(SizeOf(Key), Length(Edit2.Text)));

    // Дешифруємо...
    Decrypt_Intel AES_NIStreamECB(Source, Source.Size - Source.Position, Key,
Dest);
    Stop := GetTickCount;
    Label8.Caption := IntToStr(Stop - Start) + ' ms';
    Refresh;

    // Виводимо розшифрований текст...
    Memo_UncipherText.Text := Dest.DataString;
```

```
  finally
    Source.Free;
    Dest.Free;
  end;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
FormAbout.Show;
end;

end.
```

Файл El_Intel AES_NI.pas – шифрування/дешифрування алгоритмом _Intel AES_NI

```
unit El_Intel AES_NI;

interface

uses
  Classes, SysUtils;

type
  E_Intel AES_NIError = class(Exception);

  PInteger  = ^Integer;

  T_Intel AES_NIBuffer = array [0..15] of byte;
  T_Intel AES_NIKey128 = array [0..15] of byte;
  T_Intel AES_NIKey192 = array [0..23] of byte;
  T_Intel AES_NIKey256 = array [0..31] of byte;
  T_Intel AES_NIExpandedKey128 = array [0..43] of longword;
  T_Intel AES_NIExpandedKey192 = array [0..53] of longword;
  T_Intel AES_NIExpandedKey256 = array [0..63] of longword;

  P_Intel AES_NIBuffer =^T_Intel AES_NIBuffer;
  P_Intel AES_NIKey128 =^T_Intel AES_NIKey128;
  P_Intel AES_NIKey192 =^T_Intel AES_NIKey192;
  P_Intel AES_NIKey256 =^T_Intel AES_NIKey256;
  P_Intel AES_NIExpandedKey128 =^T_Intel AES_NIExpandedKey128;
  P_Intel AES_NIExpandedKey192 =^T_Intel AES_NIExpandedKey192;
  P_Intel AES_NIExpandedKey256 =^T_Intel AES_NIExpandedKey256;

// Розширення ключа для шифрування

procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey128;
  var ExpandedKey: T_Intel AES_NIExpandedKey128); overload;
procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey192;
  var ExpandedKey: T_Intel AES_NIExpandedKey192); overload;
procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey256;
  var ExpandedKey: T_Intel AES_NIExpandedKey256); overload;

// Блок раундів шифрування

procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey128;
  var OutBuf: T_Intel AES_NIBuffer); overload;
procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey192;
  var OutBuf: T_Intel AES_NIBuffer); overload;
procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey256;
  var OutBuf: T_Intel AES_NIBuffer); overload;

// Поток раундів Шифрування (ECB режим)

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; Dest: TStream); overload;
procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128; Dest: TStream); overload;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; Dest: TStream); overload;
procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192; Dest: TStream); overload;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; Dest: TStream); overload;
procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256; Dest: TStream); overload;
```

```
// Поток раундів шифрування (CBC режим)

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

// Перетворення сеансового ключа для дешифрування

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey128); overload;
procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey128;
  var ExpandedKey: T_Intel AES_NIExpandedKey128); overload;

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey192); overload;
procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey192;
  var ExpandedKey: T_Intel AES_NIExpandedKey192); overload;

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey256); overload;
procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey256;
  var ExpandedKey: T_Intel AES_NIExpandedKey256); overload;

// Блок раундів дешифрування

procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey128;
  var OutBuf: T_Intel AES_NIBuffer); overload;
procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey192;
  var OutBuf: T_Intel AES_NIBuffer); overload;
procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey256;
  var OutBuf: T_Intel AES_NIBuffer); overload;

// Поток раундів дешифрування (ECB режим)

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; Dest: TStream); overload;
procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128; Dest: TStream); overload;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; Dest: TStream); overload;
procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192; Dest: TStream); overload;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
```

```
  const Key: T_Intel AES_NIKey256; Dest: TStream); overload;
procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256; Dest: TStream); overload;

// Поток раундів дешифрування (CBC режим)

procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream); overload;
procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream); overload;

resourcestring
  SInvalidInBufSize = 'Не хватає розміру буферу длядешифрування';
  SReadError = 'Помилка читання потоку';
  SWriteError = 'Помилка запису потоку';

implementation

type
  PLongWord = ^LongWord;

function Min(A, B: integer): integer;
begin
  if A < B then
    Result := A
  else
    Result := B;
end;

const
  Rcon: array [1..30] of longword = (
    $00000001, $00000002, $00000004, $00000008, $00000010, $00000020,
    $00000040, $00000080, $0000001B, $00000036, $0000006C, $000000D8,
    $000000AB, $0000004D, $0000009A, $0000002F, $0000005E, $000000BC,
    $00000063, $000000C6, $00000097, $00000035, $0000006A, $000000D4,
    $000000B3, $0000007D, $000000FA, $000000EF, $000000C5, $00000091
  );

//Таблиці перестановки

  ForwardTable: array [0..255] of longword = (
    $A56363C6, $847C7CF8, $997777EE, $8D7B7BF6, $0DF2F2FF, $BD6B6BD6, $B16F6FDE,
$54C5C591,
    $50303060, $03010102, $A96767CE, $7D2B2B56, $19FEFEE7, $62D7D7B5, $E6ABAB4D,
$9A7676EC,
    $45CACA8F, $9D82821F, $40C9C989, $877D7DFA, $15FAFAEF, $EB5959B2, $C947478E,
$0BF0F0FB,
    $ECADAD41, $67D4D4B3, $FDA2A25F, $EAAFAF45, $BF9C9C23, $F7A4A453, $967272E4,
$5BC0C09B,
```

```
    $C2B7B775, $1CFDFDE1, $AE93933D, $6A26264C, $5A36366C, $413F3F7E, $02F7F7F5,
$4FCCCC83,
    $5C343468, $F4A5A551, $34E5E5D1, $08F1F1F9, $937171E2, $73D8D8AB, $53313162,
$3F15152A,
    $0C040408, $52C7C795, $65232346, $5EC3C39D, $28181830, $A1969637, $0F05050A,
$B59A9A2F,
    $0907070E, $36121224, $9B80801B, $3DE2E2DF, $26EBEBCD, $6927274E, $CDB2B27F,
$9F7575EA,
    $1B090912, $9E83831D, $742C2C58, $2E1A1A34, $2D1B1B36, $B26E6EDC, $EE5A5AB4,
$FBA0A05B,
    $F65252A4, $4D3B3B76, $61D6D6B7, $CEB3B37D, $7B292952, $3EE3E3DD, $712F2F5E,
$97848413,
    $F55353A6, $68D1D1B9, $00000000, $2CEDEDC1, $60202040, $1FFCFCE3, $C8B1B179,
$ED5B5BB6,
    $BE6A6AD4, $46CBCB8D, $D9BEBE67, $4B393972, $DE4A4A94, $D44C4C98, $E85858B0,
$4ACFCF85,
    $6BD0D0BB, $2AEFEFC5, $E5AAAA4F, $16FBFBED, $C5434386, $D74D4D9A, $55333366,
$94858511,
    $CF45458A, $10F9F9E9, $06020204, $817F7FFE, $F05050A0, $443C3C78, $BA9F9F25,
$E3A8A84B,
    $F35151A2, $FEA3A35D, $C0404080, $8A8F8F05, $AD92923F, $BC9D9D21, $48383870,
$04F5F5F1,
    $DFBCBC63, $C1B6B677, $75DADAAF, $63212142, $30101020, $1AFFFFE5, $0EF3F3FD,
$6DD2D2BF,
    $4CCDCD81, $140C0C18, $35131326, $2FECECC3, $E15F5FBE, $A2979735, $CC444488,
$3917172E,
    $57C4C493, $F2A7A755, $827E7EFC, $473D3D7A, $AC6464C8, $E75D5DBA, $2B191932,
$957373E6,
    $A06060C0, $98818119, $D14F4F9E, $7FDCDCA3, $66222244, $7E2A2A54, $AB90903B,
$8388880B,
    $CA46468C, $29EEEEC7, $D3B8B86B, $3C141428, $79DEDEA7, $E25E5EBC, $1D0B0B16,
$76DBDBAD,
    $3BE0E0DB, $56323264, $4E3A3A74, $1E0A0A14, $DB494992, $0A06060C, $6C242448,
$E45C5CB8,
    $5DC2C29F, $6ED3D3BD, $EFACAC43, $A66262C4, $A8919139, $A4959531, $37E4E4D3,
$8B7979F2,
    $32E7E7D5, $43C8C88B, $5937376E, $B76D6DDA, $8C8D8D01, $64D5D5B1, $D24E4E9C,
$E0A9A949,
    $B46C6CD8, $FA5656AC, $07F4F4F3, $25EAEACF, $AF6565CA, $8E7A7AF4, $E9AEAE47,
$18080810,
    $D5BABA6F, $887878F0, $6F25254A, $722E2E5C, $241C1C38, $F1A6A657, $C7B4B473,
$51C6C697,
    $23E8E8CB, $7CDDDDA1, $9C7474E8, $211F1F3E, $DD4B4B96, $DCBDBD61, $868B8B0D,
$858A8A0F,
    $907070E0, $423E3E7C, $C4B5B571, $AA6666CC, $D8484890, $05030306, $01F6F6F7,
$120E0E1C,
    $A36161C2, $5F35356A, $F95757AE, $D0B9B969, $91868617, $58C1C199, $271D1D3A,
$B99E9E27,
    $38E1E1D9, $13F8F8EB, $B398982B, $33111122, $BB6969D2, $70D9D9A9, $898E8E07,
$A7949433,
    $B69B9B2D, $221E1E3C, $92878715, $20E9E9C9, $49CECE87, $FF5555AA, $78282850,
$7ADFDFA5,
    $8F8C8C03, $F8A1A159, $80898909, $170D0D1A, $DABFBF65, $31E6E6D7, $C6424284,
$B86868D0,
    $C3414182, $B0999929, $772D2D5A, $110F0F1E, $CBB0B07B, $FC5454A8, $D6BBBB6D,
$3A16162C
  );

  LastForwardTable: array [0..255] of longword = (
    $00000063, $0000007C, $00000077, $0000007B, $000000F2, $0000006B, $0000006F,
$000000C5,
    $00000030, $00000001, $00000067, $0000002B, $000000FE, $000000D7, $000000AB,
$00000076,
    $000000CA, $00000082, $000000C9, $0000007D, $000000FA, $00000059, $00000047,
$000000F0,
    $000000AD, $000000D4, $000000A2, $000000AF, $0000009C, $000000A4, $00000072,
$000000C0,
    $000000B7, $000000FD, $00000093, $00000026, $00000036, $0000003F, $000000F7,
$000000CC,
```

```
    $00000034, $000000A5, $000000E5, $000000F1, $00000071, $000000D8, $00000031,
$00000015,
    $00000004, $000000C7, $00000023, $000000C3, $00000018, $00000096, $00000005,
$0000009A,
    $00000007, $00000012, $00000080, $000000E2, $000000EB, $00000027, $000000B2,
$00000075,
    $00000009, $00000083, $0000002C, $0000001A, $0000001B, $0000006E, $0000005A,
$000000A0,
    $00000052, $0000003B, $000000D6, $000000B3, $00000029, $000000E3, $0000002F,
$00000084,
    $00000053, $000000D1, $00000000, $000000ED, $00000020, $000000FC, $000000B1,
$0000005B,
    $0000006A, $000000CB, $000000BE, $00000039, $0000004A, $0000004C, $00000058,
$000000CF,
    $000000D0, $000000EF, $000000AA, $000000FB, $00000043, $0000004D, $00000033,
$00000085,
    $00000045, $000000F9, $00000002, $0000007F, $00000050, $0000003C, $0000009F,
$000000A8,
    $00000051, $000000A3, $00000040, $0000008F, $00000092, $0000009D, $00000038,
$000000F5,
    $000000BC, $000000B6, $000000DA, $00000021, $00000010, $000000FF, $000000F3,
$000000D2,
    $000000CD, $0000000C, $00000013, $000000EC, $0000005F, $00000097, $00000044,
$00000017,
    $000000C4, $000000A7, $0000007E, $0000003D, $00000064, $0000005D, $00000019,
$00000073,
    $00000060, $00000081, $0000004F, $000000DC, $00000022, $0000002A, $00000090,
$00000088,
    $00000046, $000000EE, $000000B8, $00000014, $000000DE, $0000005E, $0000000B,
$000000DB,
    $000000E0, $00000032, $0000003A, $0000000A, $00000049, $00000006, $00000024,
$0000005C,
    $000000C2, $000000D3, $000000AC, $00000062, $00000091, $00000095, $000000E4,
$00000079,
    $000000E7, $000000C8, $00000037, $0000006D, $0000008D, $000000D5, $0000004E,
$000000A9,
    $0000006C, $00000056, $000000F4, $000000EA, $00000065, $0000007A, $000000AE,
$00000008,
    $000000BA, $00000078, $00000025, $0000002E, $0000001C, $000000A6, $000000B4,
$000000C6,
    $000000E8, $000000DD, $00000074, $0000001F, $0000004B, $000000BD, $0000008B,
$0000008A,
    $00000070, $0000003E, $000000B5, $00000066, $00000048, $00000003, $000000F6,
$0000000E,
    $00000061, $00000035, $00000057, $000000B9, $00000086, $000000C1, $0000001D,
$0000009E,
    $000000E1, $000000F8, $00000098, $00000011, $00000069, $000000D9, $0000008E,
$00000094,
    $0000009B, $0000001E, $00000087, $000000E9, $000000CE, $00000055, $00000028,
$000000DF,
    $0000008C, $000000A1, $00000089, $0000000D, $000000BF, $000000E6, $00000042,
$00000068,
    $00000041, $00000099, $0000002D, $0000000F, $000000B0, $00000054, $000000BB,
$00000016
  );

  InverseTable: array [0..255] of longword = (
    $50A7F451, $5365417E, $C3A4171A, $965E273A, $CB6BAB3B, $F1459D1F, $AB58FAAC,
$9303E34B,
    $55FA3020, $F66D76AD, $9176CC88, $254C02F5, $FCD7E54F, $D7CB2AC5, $80443526,
$8FA362B5,
    $495AB1DE, $671BBA25, $980EEA45, $E1C0FE5D, $02752FC3, $12F04C81, $A397468D,
$C6F9D36B,
    $E75F8F03, $959C9215, $EB7A6DBF, $DA595295, $2D83BED4, $D3217458, $2969E049,
$44C8C98E,
    $6A89C275, $78798EF4, $6B3E5899, $DD71B927, $B64FE1BE, $17AD88F0, $66AC20C9,
$B43ACE7D,
    $184ADF63, $82311AE5, $60335197, $457F5362, $E07764B1, $84AE6BBB, $1CA081FE,
$942B08F9,
```

```
    $58684870, $19FD458F, $876CDE94, $B7F87B52, $23D373AB, $E2024B72, $578F1FE3,
$2AAB5566,
    $0728EBB2, $03C2B52F, $9A7BC586, $A50837D3, $F2872830, $B2A5BF23, $BA6A0302,
$5C8216ED,
    $2B1CCF8A, $92B479A7, $F0F207F3, $A1E2694E, $CDF4DA65, $D5BE0506, $1F6234D1,
$8AFEA6C4,
    $9D532E34, $A055F3A2, $32E18A05, $75EBF6A4, $39EC830B, $AAEF6040, $069F715E,
$51106EBD,
    $F98A213E, $3D06DD96, $AE053EDD, $46BDE64D, $B58D5491, $055DC471, $6FD40604,
$FF155060,
    $24FB9819, $97E9BDD6, $CC434089, $779ED967, $BD42E8B0, $888B8907, $385B19E7,
$DBEEC879,
    $470A7CA1, $E90F427C, $C91E84F8, $00000000, $83868009, $48ED2B32, $AC70111E,
$4E725A6C,
    $FBFF0EFD, $5638850F, $1ED5AE3D, $27392D36, $64D90F0A, $21A65C68, $D1545B9B,
$3A2E3624,
    $B1670A0C, $0FE75793, $D296EEB4, $9E919B1B, $4FC5C080, $A220DC61, $694B775A,
$161A121C,
    $0ABA93E2, $E52AA0C0, $43E0223C, $1D171B12, $0B0D090E, $ADC78BF2, $B9A8B62D,
$C8A91E14,
    $8519F157, $4C0775AF, $BBDD99EE, $FD607FA3, $9F2601F7, $BCF5725C, $C53B6644,
$347EFB5B,
    $7629438B, $DCC623CB, $68FCEDB6, $63F1E4B8, $CADC31D7, $10856342, $40229713,
$2011C684,
    $7D244A85, $F83DBBD2, $1132F9AE, $6DA129C7, $4B2F9E1D, $F330B2DC, $EC52860D,
$D0E3C177,
    $6C16B32B, $99B970A9, $FA489411, $2264E947, $C48CFCA8, $1A3FF0A0, $D82C7D56,
$EF903322,
    $C74E4987, $C1D138D9, $FEA2CA8C, $360BD498, $CF81F5A6, $28DE7AA5, $268EB7DA,
$A4BFAD3F,
    $E49D3A2C, $0D927850, $9BCC5F6A, $62467E54, $C2138DF6, $E8B8D890, $5EF7392E,
$F5AFC382,
    $BE805D9F, $7C93D069, $A92DD56F, $B31225CF, $3B99ACC8, $A77D1810, $6E639CE8,
$7BBB3BDB,
    $097826CD, $F418596E, $01B79AEC, $A89A4F83, $656E95E6, $7EE6FFAA, $08CFBC21,
$E6E815EF,
    $D99BE7BA, $CE366F4A, $D4099FEA, $D67CB029, $AFB2A431, $31233F2A, $3094A5C6,
$C066A235,
    $37BC4E74, $A6CA82FC, $B0D090E0, $15D8A733, $4A9804F1, $F7DAEC41, $0E50CD7F,
$2FF69117,
    $8DD64D76, $4DB0EF43, $544DAACC, $DF0496E4, $E3B5D19E, $1B886A4C, $B81F2CC1,
$7F516546,
    $04EA5E9D, $5D358C01, $737487FA, $2E410BFB, $5A1D67B3, $52D2DB92, $335610E9,
$1347D66D,
    $8C61D79A, $7A0CA137, $8E14F859, $893C13EB, $EE27A9CE, $35C961B7, $EDE51CE1,
$3CB1477A,
    $59DFD29C, $3F73F255, $79CE1418, $BF37C773, $EACDF753, $5BAAFD5F, $146F3DDF,
$86DB4478,
    $81F3AFCA, $3EC468B9, $2C342438, $5F40A3C2, $72C31D16, $0C25E2BC, $8B493C28,
$41950DFF,
    $7101A839, $DEB30C08, $9CE4B4D8, $90C15664, $6184CB7B, $70B632D5, $745C6C48,
$4257B8D0
  );

  LastInverseTable: array [0..255] of longword = (
    $00000052, $00000009, $0000006A, $000000D5, $00000030, $00000036, $000000A5,
$00000038,
    $000000BF, $00000040, $000000A3, $0000009E, $00000081, $000000F3, $000000D7,
$000000FB,
    $0000007C, $000000E3, $00000039, $00000082, $0000009B, $0000002F, $000000FF,
$00000087,
    $00000034, $0000008E, $00000043, $00000044, $000000C4, $000000DE, $000000E9,
$000000CB,
    $00000054, $0000007B, $00000094, $00000032, $000000A6, $000000C2, $00000023,
$0000003D,
    $000000EE, $0000004C, $00000095, $0000000B, $00000042, $000000FA, $000000C3,
$0000004E,
    $00000008, $0000002E, $000000A1, $00000066, $00000028, $000000D9, $00000024,
$000000B2,
```

```
    $00000076, $0000005B, $000000A2, $00000049, $0000006D, $0000008B, $000000D1,
$00000025,
    $00000072, $000000F8, $000000F6, $00000064, $00000086, $00000068, $00000098,
$00000016,
    $000000D4, $000000A4, $0000005C, $000000CC, $0000005D, $00000065, $000000B6,
$00000092,
    $0000006C, $00000070, $00000048, $00000050, $000000FD, $000000ED, $000000B9,
$000000DA,
    $0000005E, $00000015, $00000046, $00000057, $000000A7, $0000008D, $0000009D,
$00000084,
    $00000090, $000000D8, $000000AB, $00000000, $0000008C, $000000BC, $000000D3,
$0000000A,
    $000000F7, $000000E4, $00000058, $00000005, $000000B8, $000000B3, $00000045,
$00000006,
    $000000D0, $0000002C, $0000001E, $0000008F, $000000CA, $0000003F, $0000000F,
$00000002,
    $000000C1, $000000AF, $000000BD, $00000003, $00000001, $00000013, $0000008A,
$0000006B,
    $0000003A, $00000091, $00000011, $00000041, $0000004F, $00000067, $000000DC,
$000000EA,
    $00000097, $000000F2, $000000CF, $000000CE, $000000F0, $000000B4, $000000E6,
$00000073,
    $00000096, $000000AC, $00000074, $00000022, $000000E7, $000000AD, $00000035,
$00000085,
    $000000E2, $000000F9, $00000037, $000000E8, $0000001C, $00000075, $000000DF,
$0000006E,
    $00000047, $000000F1, $0000001A, $00000071, $0000001D, $00000029, $000000C5,
$00000089,
    $0000006F, $000000B7, $00000062, $0000000E, $000000AA, $00000018, $000000BE,
$0000001B,
    $000000FC, $00000056, $0000003E, $0000004B, $000000C6, $000000D2, $00000079,
$00000020,
    $0000009A, $000000DB, $000000C0, $000000FE, $00000078, $000000CD, $0000005A,
$000000F4,
    $0000001F, $000000DD, $000000A8, $00000033, $00000088, $00000007, $000000C7,
$00000031,
    $000000B1, $00000012, $00000010, $00000059, $00000027, $00000080, $000000EC,
$0000005F,
    $00000060, $00000051, $0000007F, $000000A9, $00000019, $000000B5, $0000004A,
$0000000D,
    $0000002D, $000000E5, $0000007A, $0000009F, $00000093, $000000C9, $0000009C,
$000000EF,
    $000000A0, $000000E0, $0000003B, $0000004D, $000000AE, $0000002A, $000000F5,
$000000B0,
    $000000C8, $000000EB, $000000BB, $0000003C, $00000083, $00000053, $00000099,
$00000061,
    $00000017, $0000002B, $00000004, $0000007E, $000000BA, $00000077, $000000D6,
$00000026,
    $000000E1, $00000069, $00000014, $00000063, $00000055, $00000021, $0000000C,
$0000007D
  );

//Розширення ключа для шифрування

procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey128;
var ExpandedKey: T_Intel AES_NIExpandedKey128);
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 3] shl 24) or (ExpandedKey[I + 3] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
```

```
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 4] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
    Inc(J);
    ExpandedKey[I + 5] := ExpandedKey[I + 1] xor ExpandedKey[I + 4];
    ExpandedKey[I + 6] := ExpandedKey[I + 2] xor ExpandedKey[I + 5];
    ExpandedKey[I + 7] := ExpandedKey[I + 3] xor ExpandedKey[I + 6];
    Inc(I, 4);
  until I >= 40;
end;

procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey192;
var ExpandedKey: T_Intel AES_NIExpandedKey192); overload;
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  ExpandedKey[4] := PLongWord(@Key[16])^;
  ExpandedKey[5] := PLongWord(@Key[20])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 5] shl 24) or (ExpandedKey[I + 5] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 6] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
    Inc(J);
    ExpandedKey[I + 7] := ExpandedKey[I + 1] xor ExpandedKey[I + 6];
    ExpandedKey[I + 8] := ExpandedKey[I + 2] xor ExpandedKey[I + 7];
    ExpandedKey[I + 9] := ExpandedKey[I + 3] xor ExpandedKey[I + 8];
    ExpandedKey[I + 10] := ExpandedKey[I + 4] xor ExpandedKey[I + 9];
    ExpandedKey[I + 11] := ExpandedKey[I + 5] xor ExpandedKey[I + 10];
    Inc(I, 6);
  until I >= 46;
end;

procedure Expand_Intel AES_NIKeyForEncryption(const Key: T_Intel AES_NIKey256;
var ExpandedKey: T_Intel AES_NIExpandedKey256); overload;
var
  I, J: integer;
  T: longword;
  W0, W1, W2, W3: longword;
begin
  ExpandedKey[0] := PLongWord(@Key[0])^;
  ExpandedKey[1] := PLongWord(@Key[4])^;
  ExpandedKey[2] := PLongWord(@Key[8])^;
  ExpandedKey[3] := PLongWord(@Key[12])^;
  ExpandedKey[4] := PLongWord(@Key[16])^;
  ExpandedKey[5] := PLongWord(@Key[20])^;
  ExpandedKey[6] := PLongWord(@Key[24])^;
  ExpandedKey[7] := PLongWord(@Key[28])^;
  I := 0; J := 1;
  repeat
    T := (ExpandedKey[I + 7] shl 24) or (ExpandedKey[I + 7] shr 8);
    W0 := LastForwardTable[Byte(T)]; W1 := LastForwardTable[Byte(T shr 8)];
    W2 := LastForwardTable[Byte(T shr 16)]; W3 := LastForwardTable[Byte(T shr
24)];
    ExpandedKey[I + 8] := ExpandedKey[I] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8))) xor Rcon[J];
```

```
    Inc(J);
    ExpandedKey[I + 9] := ExpandedKey[I + 1] xor ExpandedKey[I + 8];
    ExpandedKey[I + 10] := ExpandedKey[I + 2] xor ExpandedKey[I + 9];
    ExpandedKey[I + 11] := ExpandedKey[I + 3] xor ExpandedKey[I + 10];
    W0 := LastForwardTable[Byte(ExpandedKey[I + 11])];
    W1 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 8)];
    W2 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 16)];
    W3 := LastForwardTable[Byte(ExpandedKey[I + 11] shr 24)];
    ExpandedKey[I + 12] := ExpandedKey[I + 4] xor
      (W0 xor ((W1 shl 8) or (W1 shr 24)) xor
      ((W2 shl 16) or (W2 shr 16)) xor ((W3 shl 24) or (W3 shr 8)));
    ExpandedKey[I + 13] := ExpandedKey[I + 5] xor ExpandedKey[I + 12];
    ExpandedKey[I + 14] := ExpandedKey[I + 6] xor ExpandedKey[I + 13];
    ExpandedKey[I + 15] := ExpandedKey[I + 7] xor ExpandedKey[I + 14];
    Inc(I, 8);
  until I >= 52;
end;



//Процедура шифрування

procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey128;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // Ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація - 9 раз
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;


procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey192;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація - 11 раз
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
```

```
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 10
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 11
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure Encrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey256;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[0];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[1];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[2];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[3];
  // Попередня трансформація 13 разів
  // раунд 1
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // раунд 2
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 3
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 4
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
```

```
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 5
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 9
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 10
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 11
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
```

```
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 12
  W0 := ForwardTable[Byte(T1[0])]; W1 := ForwardTable[Byte(T1[1] shr 8)];
  W2 := ForwardTable[Byte(T1[2] shr 16)]; W3 := ForwardTable[Byte(T1[3] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := ForwardTable[Byte(T1[1])]; W1 := ForwardTable[Byte(T1[2] shr 8)];
  W2 := ForwardTable[Byte(T1[3] shr 16)]; W3 := ForwardTable[Byte(T1[0] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := ForwardTable[Byte(T1[2])]; W1 := ForwardTable[Byte(T1[3] shr 8)];
  W2 := ForwardTable[Byte(T1[0] shr 16)]; W3 := ForwardTable[Byte(T1[1] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := ForwardTable[Byte(T1[3])]; W1 := ForwardTable[Byte(T1[0] shr 8)];
  W2 := ForwardTable[Byte(T1[1] shr 16)]; W3 := ForwardTable[Byte(T1[2] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // раунд 13
  W0 := ForwardTable[Byte(T0[0])]; W1 := ForwardTable[Byte(T0[1] shr 8)];
  W2 := ForwardTable[Byte(T0[2] shr 16)]; W3 := ForwardTable[Byte(T0[3] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[52];
  W0 := ForwardTable[Byte(T0[1])]; W1 := ForwardTable[Byte(T0[2] shr 8)];
  W2 := ForwardTable[Byte(T0[3] shr 16)]; W3 := ForwardTable[Byte(T0[0] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[53];
  W0 := ForwardTable[Byte(T0[2])]; W1 := ForwardTable[Byte(T0[3] shr 8)];
  W2 := ForwardTable[Byte(T0[0] shr 16)]; W3 := ForwardTable[Byte(T0[1] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[54];
  W0 := ForwardTable[Byte(T0[3])]; W1 := ForwardTable[Byte(T0[0] shr 8)];
  W2 := ForwardTable[Byte(T0[1] shr 16)]; W3 := ForwardTable[Byte(T0[2] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[55];
  // останій раунд перетворень
  W0 := LastForwardTable[Byte(T1[0])]; W1 := LastForwardTable[Byte(T1[1] shr
8)];
  W2 := LastForwardTable[Byte(T1[2] shr 16)]; W3 := LastForwardTable[Byte(T1[3]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[56];
```

```
  W0 := LastForwardTable[Byte(T1[1])]; W1 := LastForwardTable[Byte(T1[2] shr
8)];
  W2 := LastForwardTable[Byte(T1[3] shr 16)]; W3 := LastForwardTable[Byte(T1[0]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[57];
  W0 := LastForwardTable[Byte(T1[2])]; W1 := LastForwardTable[Byte(T1[3] shr
8)];
  W2 := LastForwardTable[Byte(T1[0] shr 16)]; W3 := LastForwardTable[Byte(T1[1]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[58];
  W0 := LastForwardTable[Byte(T1[3])]; W1 := LastForwardTable[Byte(T1[0] shr
8)];
  W2 := LastForwardTable[Byte(T1[1] shr 16)]; W3 := LastForwardTable[Byte(T1[2]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[59];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;
//Розширення ключа для дешифрування

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey128);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 9 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
```

```
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey128;
var ExpandedKey: T_Intel AES_NIExpandedKey128);
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Expand_Intel AES_NIKeyForDecryption(ExpandedKey);
end;

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey192);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 11 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
```

```
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey192;
var ExpandedKey: T_Intel AES_NIExpandedKey192);
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Expand_Intel AES_NIKeyForDecryption(ExpandedKey);
end;

procedure Expand_Intel AES_NIKeyForDecryption(var ExpandedKey: T_Intel
AES_NIExpandedKey256);
var
  I: integer;
  U, F2, F4, F8, F9: longword;
begin
  for I := 1 to 13 do
  begin
    F9 := ExpandedKey[I * 4];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 1];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 1] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 2];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 2] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
    F9 := ExpandedKey[I * 4 + 3];
    U := F9 and $80808080;
    F2 := ((F9 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F2 and $80808080;
    F4 := ((F2 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
    U := F4 and $80808080;
    F8 := ((F4 and $7F7F7F7F) shl 1) xor ((U - (U shr 7)) and $1B1B1B1B);
```

```
    F9 := F9 xor F8;
    ExpandedKey[I * 4 + 3] := F2 xor F4 xor F8 xor
      (((F2 xor F9) shl 24) or ((F2 xor F9) shr 8)) xor
      (((F4 xor F9) shl 16) or ((F4 xor F9) shr 16)) xor ((F9 shl 8) or (F9 shr
24));
  end;
end;

procedure Expand_Intel AES_NIKeyForDecryption(const Key: T_Intel AES_NIKey256;
var ExpandedKey: T_Intel AES_NIExpandedKey256);
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Expand_Intel AES_NIKeyForDecryption(ExpandedKey);
end;

//Процедура дешифрування

procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey128;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[40];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[41];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[42];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[43];
  // Попередня трансформація  9 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey192;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[48];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[49];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[50];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[51];
  // Попередня трансформація  11 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 10
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 11
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

procedure Decrypt_Intel AES_NI(const InBuf: T_Intel AES_NIBuffer; const Key:
T_Intel AES_NIExpandedKey256;
  var OutBuf: T_Intel AES_NIBuffer);
var
  T0, T1: array [0..3] of longword;
  W0, W1, W2, W3: longword;
begin
  // ініціалізація
  T0[0] := PLongWord(@InBuf[0])^ xor Key[56];
  T0[1] := PLongWord(@InBuf[4])^ xor Key[57];
  T0[2] := PLongWord(@InBuf[8])^ xor Key[58];
  T0[3] := PLongWord(@InBuf[12])^ xor Key[59];
  // Попередня трансформація  13 разів
  // раунд 1
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[52];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[53];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[54];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[55];
  // раунд 2
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[48];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[49];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[50];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[51];
  // раунд 3
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[44];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[45];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[46];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[47];
  // раунд 4
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[40];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[41];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[42];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[43];
  // раунд 5
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[36];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[37];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[38];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[39];
  // раунд 6
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[32];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[33];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[34];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[35];
  // раунд 7
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[28];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[29];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[30];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[31];
  // раунд 8
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
```

```
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[24];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[25];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[26];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[27];
  // раунд 9
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[20];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[21];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[22];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[23];
  // раунд 10
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[16];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[17];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[18];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[19];
  // раунд 11
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[12];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
```

```
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[13];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[14];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[15];
  // раунд 12
  W0 := InverseTable[Byte(T1[0])]; W1 := InverseTable[Byte(T1[3] shr 8)];
  W2 := InverseTable[Byte(T1[2] shr 16)]; W3 := InverseTable[Byte(T1[1] shr
24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[8];
  W0 := InverseTable[Byte(T1[1])]; W1 := InverseTable[Byte(T1[0] shr 8)];
  W2 := InverseTable[Byte(T1[3] shr 16)]; W3 := InverseTable[Byte(T1[2] shr
24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[9];
  W0 := InverseTable[Byte(T1[2])]; W1 := InverseTable[Byte(T1[1] shr 8)];
  W2 := InverseTable[Byte(T1[0] shr 16)]; W3 := InverseTable[Byte(T1[3] shr
24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[10];
  W0 := InverseTable[Byte(T1[3])]; W1 := InverseTable[Byte(T1[2] shr 8)];
  W2 := InverseTable[Byte(T1[1] shr 16)]; W3 := InverseTable[Byte(T1[0] shr
24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[11];
  // раунд 13
  W0 := InverseTable[Byte(T0[0])]; W1 := InverseTable[Byte(T0[3] shr 8)];
  W2 := InverseTable[Byte(T0[2] shr 16)]; W3 := InverseTable[Byte(T0[1] shr
24)];
  T1[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[4];
  W0 := InverseTable[Byte(T0[1])]; W1 := InverseTable[Byte(T0[0] shr 8)];
  W2 := InverseTable[Byte(T0[3] shr 16)]; W3 := InverseTable[Byte(T0[2] shr
24)];
  T1[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[5];
  W0 := InverseTable[Byte(T0[2])]; W1 := InverseTable[Byte(T0[1] shr 8)];
  W2 := InverseTable[Byte(T0[0] shr 16)]; W3 := InverseTable[Byte(T0[3] shr
24)];
  T1[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[6];
  W0 := InverseTable[Byte(T0[3])]; W1 := InverseTable[Byte(T0[2] shr 8)];
  W2 := InverseTable[Byte(T0[1] shr 16)]; W3 := InverseTable[Byte(T0[0] shr
24)];
  T1[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[7];
  // останій раунд перетворень
  W0 := LastInverseTable[Byte(T1[0])]; W1 := LastInverseTable[Byte(T1[3] shr
8)];
  W2 := LastInverseTable[Byte(T1[2] shr 16)]; W3 := LastInverseTable[Byte(T1[1]
shr 24)];
  T0[0] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[0];
  W0 := LastInverseTable[Byte(T1[1])]; W1 := LastInverseTable[Byte(T1[0] shr
8)];
  W2 := LastInverseTable[Byte(T1[3] shr 16)]; W3 := LastInverseTable[Byte(T1[2]
shr 24)];
  T0[1] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[1];
```

```
  W0 := LastInverseTable[Byte(T1[2])]; W1 := LastInverseTable[Byte(T1[1] shr
8)];
  W2 := LastInverseTable[Byte(T1[0] shr 16)]; W3 := LastInverseTable[Byte(T1[3]
shr 24)];
  T0[2] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[2];
  W0 := LastInverseTable[Byte(T1[3])]; W1 := LastInverseTable[Byte(T1[2] shr
8)];
  W2 := LastInverseTable[Byte(T1[1] shr 16)]; W3 := LastInverseTable[Byte(T1[0]
shr 24)];
  T0[3] := (W0 xor ((W1 shl 8) or (W1 shr 24)) xor ((W2 shl 16) or (W2 shr 16))
    xor ((W3 shl 24) or (W3 shr 8))) xor Key[3];
  // кінець роботи алгоритму
  PLongWord(@OutBuf[0])^ := T0[0]; PLongWord(@OutBuf[4])^ := T0[1];
  PLongWord(@OutBuf[8])^ := T0[2]; PLongWord(@OutBuf[12])^ := T0[3];
end;

// Поток раундів шифрування (ECB режим)

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey128;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey192;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey256;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
```

```
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192; Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure Encrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
```

```
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

// Поток раундів дешифрування (ECB режим)

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey128;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey192;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192; Dest: TStream);
```

```
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; Dest: TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey256;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure Decrypt_Intel AES_NIStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;

// Поток раундів шифрування (CBC режим)

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
```

```
  ExpandedKey: T_Intel AES_NIExpandedKey128;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey192;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
```

```
  const ExpandedKey: T_Intel AES_NIExpandedKey192;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey256;
begin
  Expand_Intel AES_NIKeyForEncryption(Key, ExpandedKey);
  Encrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure Encrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
```

```
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  Vector := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
  if Count > 0 then
  begin
    Done := Source.Read(TempIn, Count);
    if Done < Count then
      raise EStreamError.Create(SReadError);
    FillChar(TempIn[Count], SizeOf(TempIn) - Count, 0);
    PLongWord(@TempIn[0])^ := PLongWord(@TempIn[0])^ xor PLongWord(@Vector[0])^;
    PLongWord(@TempIn[4])^ := PLongWord(@TempIn[4])^ xor PLongWord(@Vector[4])^;
    PLongWord(@TempIn[8])^ := PLongWord(@TempIn[8])^ xor PLongWord(@Vector[8])^;
    PLongWord(@TempIn[12])^ := PLongWord(@TempIn[12])^ xor
PLongWord(@Vector[12])^;
    Encrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

// Поток раундів дешифрування (CBC режим)

procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey128; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey128;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey128;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Vector1, Vector2: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
```

```
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;


procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey192; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey192;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;


procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey192;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Vector1, Vector2: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do


begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
```

```
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;


procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const Key: T_Intel AES_NIKey256; const InitVector: T_Intel AES_NIBuffer; Dest:
TStream);
var
  ExpandedKey: T_Intel AES_NIExpandedKey256;
begin
  Expand_Intel AES_NIKeyForDecryption(Key, ExpandedKey);
  Decrypt_Intel AES_NIStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;


procedure Decrypt_Intel AES_NIStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: T_Intel AES_NIExpandedKey256;  const InitVector: T_Intel
AES_NIBuffer;
  Dest: TStream);
var
  TempIn, TempOut: T_Intel AES_NIBuffer;
  Vector1, Vector2: T_Intel AES_NIBuffer;
  Done: cardinal;
begin
  if Count = 0 then
  begin
    Source.Position := 0;
    Count := Source.Size;
  end
  else Count := Min(Count, Source.Size - Source.Position);
  if Count = 0 then exit;
  if (Count mod SizeOf(T_Intel AES_NIBuffer)) > 0 then
    raise E_Intel AES_NIError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(T_Intel AES_NIBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    Decrypt_Intel AES_NI(TempIn, ExpandedKey, TempOut);
    PLongWord(@TempOut[0])^ := PLongWord(@TempOut[0])^ xor
PLongWord(@Vector1[0])^;
    PLongWord(@TempOut[4])^ := PLongWord(@TempOut[4])^ xor
PLongWord(@Vector1[4])^;
    PLongWord(@TempOut[8])^ := PLongWord(@TempOut[8])^ xor
PLongWord(@Vector1[8])^;
    PLongWord(@TempOut[12])^ := PLongWord(@TempOut[12])^ xor
PLongWord(@Vector1[12])^;
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError(SWriteError);
    Vector1 := Vector2;
    Dec(Count, SizeOf(T_Intel AES_NIBuffer));
  end;
end;

end.
```

Файл about.pas - довідка

```
unit about;

interface

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

type
  TForm5 = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.dfm}

procedure TForm5.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 захищеного
документообігу з використанням інструкцій AES процесора Intel  CORE I9-13900K
S1700');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Керівник: Коваленко О.В.');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Розробив: студент Босенко Денис Сергійович');
Memo1.Lines.Add('                   гр. КБ-21-3СК');
Memo1.Lines.Add('');
Memo1.Lines.Add('Кропивницький 2024');
Memo1.Lines.Add('');
end;

procedure TForm5.Button1Click(Sender: TObject);
begin
Form5.Close;
end;
end.
```