Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 123 *"Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

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

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

Жупилі Миколі Миколайовичі
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів*

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

затверджені наказом вищого навчального закладу № 20-13 від 17.08.2022 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Жупило М.М.
(прізвище та ініціали)

АНОТАЦІЯ

Жупило М.М. Дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

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

Метою розробки є дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

Об'єктом дослідження є процес забезпечення конфіденційності даних хмарних сервісів.

Предметом дослідження є методи забезпечення конфіденційності даних хмарних сервісів.

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

Результат роботи – програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

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

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

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

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерна інженерія, конфіденційність даних, хмарні сервіси

ABSTRACT

Zhupylo M.M. Research and software implementation of the data privacy protection system of cloud services. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this final qualification work for the second (master's) level of higher education, software is developed, which is intended for the system of ensuring the confidentiality of data of cloud services.

The purpose of the development is the research and software implementation of the system for ensuring the confidentiality of data of cloud services.

The object of the study is the process of ensuring the confidentiality of data in cloud services.

The subject of the study is the methods of ensuring the confidentiality of data of cloud services.

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

The result of the work is the software implementation of the system for ensuring the confidentiality of data of cloud services.

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

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

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

The program was developed in the Delphi 10.4 environment.

Keywords: computer engineering, data privacy, cloud services

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. Розвиток нових інформаційних технологій і загальна комп'ютеризація привели до того, що інформаційна безпека не тільки стає обов'язковою, вона ще й одна з характеристик інформаційної системи. Існує досить великий клас систем обробки інформації, при розробці яких фактор безпеки відіграє першорядну роль (наприклад, банківські інформаційні системи). До цього класу систем відносяться й різноманітні хмарні сервіси.

Під безпекою інформаційної системи розуміється захищеність системи від випадкового або навмисного втручання в нормальний процес її функціонування, від спроб розкрадання (несанкціонованого одержання) інформації, модифікації або фізичного руйнування її компонентів. Інакше кажучи, це здатність протидіяти різним впливам, що обурюють, на інформаційної системи.

Під погрозою безпеки інформації розуміються події або дії, які можуть привести до перекручування, несанкціонованого використання або навіть до руйнування інформаційних ресурсів керованої системи, а також програмних і апаратних засобів.

У своїх протиправних діях, спрямованих на оволодіння чужими секретами, взламники прагнуть знайти такі джерела конфіденційної інформації, які б давали їм найбільш достовірну інформацію в максимальних обсягах з мінімальними витратами на її одержання. За допомогою різного роду вивертів і безлічі прийомів і засобів підбираються шляхи й підходи до таких джерел. У цьому випадку під джерелом інформації мається на увазі матеріальний об'єкт, що володіє певними відомостями, що представляють конкретний інтерес для зловмисників або конкурентів.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

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

– Огляд існуючих систем забезпечення конфіденційності даних хмарних сервісів.

– Дослідження системи забезпечення конфіденційності даних хмарних сервісів.

– Програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

Об'єктом дослідження є процес забезпечення конфіденційності даних хмарних сервісів.

Предметом дослідження є методи забезпечення конфіденційності даних хмарних сервісів.

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

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

– Удосконалено метод забезпечення конфіденційності даних хмарних сервісів.

– Розроблено вітчизняний продукт забезпечення конфіденційності даних хмарних сервісів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі забезпечення конфіденційності даних хмарних сервісів.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмний продукт розроблювальний, у результаті виконання магістерського проектування, призначається для забезпечення конфіденційності даних хмарних сервісів, за рахунок застосування алгоритму AES. Цей алгоритм обраний виходячи з наступних передумов. Алгоритм DES (Data Encryption Standard), розроблений корпорацією IBM і був з 1977 федеральним стандартом шифрування даних США, використовувався для шифрування не тільки урядом США, але й одержав широке поширення по усьому світі серед приватних користувачів. З ростом обчислювальної потужності комп'ютерів стали виникати питання про криптостійкості DES'a перед розкриттям методом "грубої сили", але стандарт успішно проходив повторні сертифікації, що проводилися в 1983, 1988 і 1993. Хоча до середини 90-х років стала очевидним невідповідність загальноприйнятого стандарту шифрування DES (Data Encryption Standard) сучасним вимогам. У першу чергу з-за недостатньої довжини ключа всього в 56 біт. По даним Брюса Шнайера вже в 1993 році існували пристрої здатні розкрити DES за прийнятний час.

Процес розробки нового федерального інформаційного стандарту (FIPS) для шифрування даних Advanced Encryption Standard (AES) був ініційований Національним Інститутом стандартів і технологій (NIST). На початку січня 1997 року NIST оголосив про початок розробки AES, випустивши документ "Announcing development of a federal information processing standard for advanced encryption standard", що містить первинні вимоги до алгоритму [2]:

– Алгоритм шифрування AES повинен бути відкрито опублікований.

– Алгоритм повинен бути симетричним блоковим шифром.

– AES повинен передбачати можливість збільшення довжини ключа.

– AES повинен бути легко реалізуємо й апаратної й програмно.

– AES повинен поширюватися безкоштовно або бути загальнодоступним у рамках патентів ANSI.

У вересні 1997 року вийшов уточнюючий документ "Call for AES Candidate Algorithms", що повідомляв про проведення конкурсу на AES і утримуючу офіційну вимоги до кандидатів. Зокрема алгоритм повинен підтримувати наступні комбінації довжин блоку й ключа 128 – 128, 128 – 192 і 128 – 256 бітів. До 15 червня 1998 року був заявлений 21 криптографічний алгоритм, але тільки 15 з яких задовольняли первісним вимогам [3].

Таблиця 1.1 – Алгоритми-претенденти

Країна	Алгоритм	Автори
Australia	LOKI97	Lawrie Brown, Josef Pieprzyk, Jennifer Seberry
Belgium	RIJNDAEL	Joan Daemen, Vincent Rijmen
Canada	CAST– 256	Entrust Technologies, Inc
	DEAL	Richard Outerbridge, Lars Knudsen
Costa Rica	FROG	TecApro Internacional S.A.
France	DFC	Centre National pour la Recherche Scientifique
Germany	MAGENTA	Deutsche Telekom AG
Japan	E2	Nippon Telegraph and Telephone Corporation
Korea	CRYPTON	Future Systems, Inc
USA	HPC	Rich Schroeppel
	MARS	IBM
	RC6	RSA Laboratories
	SAFER+	Cylink Corporation
	TWOFISH	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson
UK, Israel, Norway	SERPENT	Ross Anderson, Eli Biham, Lars Knudsen

Наприкінці серпня 1998 року в Каліфорнії відбулася конференція, присвячена відбору кандидатів на AES, що одержала назву AES1. На конференції давалися короткі описи алгоритмів шифрування, і були дані відповіді на накопичені питання.

Як основні критерії оцінки алгоритмів були [3]:

– Криптостійкость. Проводилося порівняння алгоритмів на ступінь незалежності вихідного блоку від випадкової перестановки бітів вхідного блоку, схильність відомим криптоатакам. Кожний кандидат повинен був представити оцінну криптостійкость алгоритму.

– Вартість. Кандидат надавав інформацію про ліцензійні угоди й патенти на алгоритм. Також ураховувалася продуктивність алгоритму (швидкість шифрування), необхідний для шифрування розмір пам'яті.

– Особливості алгоритму і його реалізації. Гнучкість, апаратна й програмна зручність реалізації.

У квітні 1999 року в Римі відбулася конференція AES2, у рамках якої проводилися порівняння продуктивності програмних реалізацій алгоритмів з оптимизациями під мови C и Java. Найбільшу швидкість шифрування/дешифрування показав алгоритм Crypton (40 Мбайт/с), найменшу Magenta і HPC (2Мбайт/с). Хоча при проведенні тестів на різних платформах і з різними компіляторами, отримані результати досить сильно розрізнялися. Всі блокові алгоритми можна розбити на дві основні групи:

– мережі, що використовують, Фейстеля;

– мережі перестановок/підстановок (SP-мережі) засновані на послідовних перестановках і підстановках, що залежать від ключа.

Серед алгоритмів-претендентів до першого відносяться CAST-256, DEAL, DFC, E2, LOKI97, MAGENTA, MARS, RC6, TWOFISH, до других CRYPTON, Rijndael, SAFER+ і SERPENT. Алгоритми FROG і HPC не підпадали ні під одну з категорій, але вході обговорення кандидатів не виявлено яких-небудь видатних якостей даних алгоритмів.

Результати проробленої аналітиками роботи з вивчення алгоритмів-фіналістів NIST сформулював у вигляді звіту. Даний звіт містить як результати аналізу алгоритмів, так і обґрунтування критеріїв, по яких виконувалася оцінка.

Таблиця 1.2 – Порівняльні оцінки алгоритмів-фіналістів

№	Категорія	Serpent	Twofish	MARS	RC6	Rijndael
1	Криптостійкість	+	+	+	+	+
2	Запас криптостійкості	++	++	++	+	+
3	Швидкість шифрування при програмній реалізації	-	±	±	+	+
4	Швидкість розширення ключа при програмній реалізації	±	-	±	±	+
5	Смарт-карти з великим обсягом ресурсів	+	+	-	±	++
6	Смарт-карти з обмеженим обсягом ресурсів	±	+	-	±	++
7	Апаратна реалізація (ПЛІС)	+	+	-	±	+
8	Апаратна реалізація (спеціалізована мікросхема)	+	±	-	-	+
9	Захист від атак за часом виконання й споживаної потужності	+	±	-	-	+
10	Захист від атак по споживаній потужності на процедуру розширення ключа	±	±	±	±	-
11	Захист від атак по споживаній потужності на реалізації в смарт-картах	±	+	-	±	+
12	Можливість розширення ключа «на льоту»	+	+	±	±	±
13	Наявність варіантів реалізації (без втрат у сумісності)	+	+	±	±	+
14	Можливість паралельних обчислень	±	±	±	±	+

Варто прокоментувати деякі рядки наведеної таблиці:

1. Криптостійкість всіх алгоритмів-фіналістів виявилася достатньою – у процесі досліджень не було виявлено яких-небудь реально реалізованих атак на повноцінні й повнораундові версії алгоритмів. У цьому випадку криптоаналітики звичайно досліджують варіанти алгоритмів з усіченим числом раундів, або з деякими внесеними змінами, незначними, але ослабляючими алгоритм. Під запасом криптостійкості (security margin) експерти NIST мають на увазі співвідношення повного (передбаченого в специфікаціях алгоритмів) числа раундів і максимального з тих варіантів, проти яких діють які-небудь криптоаналитичні атаки. Наприклад, за допомогою диференційно-лінійного криптоанализа розкривається 11-раундовий Serpent, тоді як в оригінальному алгоритмі виконується 32 раунду. Експерти NIST у звіті попередили, що дана оцінка є досить поверхневою й не може бути значимою при виборі алгоритму-переможця конкурсу, але, проте, відзначили, що запас криптостійкості в Rijndael і RC6 трохи нижче, ніж в інших алгоритмів-фіналістів.

2. У пп. 5-8 наведена порівняльна оцінка можливості й ефективності реалізації алгоритмів у перерахованих пристроях.

3. У пп. 9-11 мається на увазі, наскільки операції, виконувані конкретним алгоритмом, можуть бути піддані аналізу зазначеним методом. При цьому приймалося в розрахунок те, що операції можуть бути модифіковані з метою ускладнення криптоаналіза за рахунок втрати у швидкості шифрування (наприклад, проблемне в цьому змісті обертання на змінне число біт може примусово виконуватися за рівне число тактів – тобто максимально можливе для даної операції; саме подібні міри протидії атакам за часом виконання й споживаної потужності рекомендує їхній винахідник Піл Кохер (Paul C. Kocher).

4. З описів алгоритмів видно, що всі вони підтримують розширення ключа «на льоту» (тобто підключи можуть генеруватися безпосередньо в процесі шифрування – у міру необхідності), однак, тільки Serpent і Twofish підтримують таку можливість без яких-небудь обмежень.

5. Під наявністю варіантів реалізації (implementation flexibility) мається на увазі можливість різним образом реалізовувати які-небудь операції алгоритму з оптимізацією під конкретні цілі. Найбільш показовими в цьому змісті є згадані раніше варіанти процедури розширення ключа алгоритму Twofish, що дозволяють оптимізувати реалізацію алгоритму в залежності, насамперед, від частоти зміни ключа.

Таблиця 1.3 – Основні переваги й недоліки алгоритмів-фіналістів

Алгоритм	Переваги	Недоліки
Serpent	1. Проста структура алгоритму полегшує його аналіз із метою знаходження можливих уразливістей. 2. Serpent ефективно реалізуємо апаратно й в умовах обмежених ресурсів. 3. Serpent легко модифікується з метою захисту від атак за часом виконання й споживаної потужності (однак, за рахунок зниження швидкості).	1. Є самим повільним з алгоритмів-фіналістів у програмних реалізаціях. 2. Процедури зашифрування й розшифрування абсолютно різні, тобто вимагають роздільної реалізації. 3. Розпаралелювання обчислень при шифруванні алгоритмом Serpent реалізовано з обмеженнями.
Twofish	1. Twofish ефективно реалізуємо апаратно й в умовах обмежених ресурсів. 2. Зашифрування й розшифрування в алгоритмі Twofish практично ідентичні.	1. Складність структури алгоритму утрудняє його аналіз. 2. Складна й повільна процедура розширення ключа. 3. Відносно складно захищається від атак за часом виконання й споживаної потужності.

Продовження таблиці 1.3

Алгоритм	Переваги	Недоліки
Twofish	3. Є кращим з алгоритмів-фіналістів з погляду підтримки розширення ключа «на льоту». 4. Кілька варіантів реалізації дозволяють оптимізувати алгоритм для конкретних застосувань.	4. Розпаралелювання обчислень при шифруванні алгоритмом Twofish реалізовано з обмеженнями.
MARS	Зашифрування й розшифрування в алгоритмі MARS практично ідентичні.	1. Винятково складна структура алгоритму з раундами різних типів утрудняє як аналіз алгоритму, так і його реалізацію. 2. Виникають проблеми при програмній реалізації на тих платформах, які не підтримують 32-бітне множення й обертання на змінне число біт. 3. Алгоритм MARS не може бути ефективно реалізований апаратно й в умовах обмежених ресурсів. 4. Складно захищається від атак за часом виконання й споживаної потужності. 5. MARS гірше інших алгоритмів-фіналістів підтримує розширення ключів «на льоту». 6. Розпаралелювання обчислень при шифруванні алгоритмом MARS реалізовано з обмеженнями.

Продовження таблиці 1.3

Алгоритм	Переваги	Недоліки
RC6	1. Проста структура алгоритму полегшує його аналіз. Крім того, алгоритм успадкував частину перетворень від свого попередника – алгоритму RC5, ретельно проаналізованого до конкурсу AES. 2. Найшвидший з алгоритмів-фіналістів на 32-бітних платформах. 3. Зашифрування й розшифрування в алгоритмі RC6 практично ідентичні.	1. Швидкість шифрування при програмній реалізації сильно залежить від того, чи підтримує платформа 32-бітне множення й обертання на змінне число біт. 2. RC6 складно реалізуємо апаратно й в умовах обмежених ресурсів. 3. Досить складно захищається від атак за часом виконання й споживаної потужності. 4. Недостатньо повно підтримує розширення ключів «на льоту». 5. Розпаралелювання обчислень при шифруванні алгоритмом RC6 реалізовано з обмеженнями.

У результаті первинного обговорення була виявлена слабка криптостійкость алгоритму MAGENTA, незабаром з'явилися дані по криптоанализу алгоритмів FROG, LOKI, що показують слабості алгоритмів щодо інших алгоритмів. Також частина алгоритмів мали низькі шанси на успіх з-за вкрай малої швидкості шифрування/дешифрування. Відбір фіналістів за названими критеріями тривав до початку серпня 1999р. 9серпня NIST випустила прес-реліз "Announces AES Finalists", у якому оголошувалося про п'ять фіналістів: MARS, RC6, Rijndael, Serpent, TwoFish.

Як відомо, проаналізувавши результати всіх досліджень, експерти вибрали як стандарт AES алгоритм Rijndael. Що не виглядає дивним – у таблиці з характеристиками алгоритмів чітко видно, що практично по всіх характеристиках Rindael, як мінімум, не уступає іншим алгоритмам-фіналістам.

Що стосується алгоритмів Serpent, Twofish, MARS і RC6, те видно, що вони практично рівнозначні по сукупності характеристик, за винятком алгоритму MARS, що має істотно більше недоліків, у тому числі, алгоритм практично не реалізуємий в умовах обмежених ресурсів.

1.2 Область застосування

Областю застосування є захист даних на носіях інформації. Побудова будь-якої системи зберігання даних повинне передбачати рішення проблеми забезпечення гарантованого захисту даних про конкретні особи. Засобу захисту інформації повинні забезпечувати:

– захист інформації від несанкціонованої модифікації й руйнування на всіх етапах її обробки, зберігання й передачі;

– автентифікацію сторін, що роблять обмін інформацією (підтвердження дійсності відправника й одержувача);

– розмежування прав користувачів і обслуговуючого персоналу при доступі до інформаційних ресурсів системи зберігання даних, а також при зберіганні й наданні конфіденційної інформації, у тому числі захист від несанкціонованого доступу користувачів відомчих інформаційних систем до інформаційних ресурсів системи зберігання даних;

– можливість доказу неправомочності дій користувачів і обслуговуючого персоналу системи зберігання даних;

– захист інформації від несанкціонованого доступу засобами перевірки повноважень користувачів і обслуговуючого персоналу на використання інформаційних ресурсів системи зберігання даних (можливість несанкціонованої зміни або знищення цієї інформації, як і несанкціоноване одержання, зміна або знищення інформації про громадян третіми особами повинні бути виключені);

– захист від несанкціонованої модифікації програмного забезпечення, включаючи захист від впровадження “вірусів” у програмні продукти;

– захист інформації від випадкових руйнувань;

– дублювання інформації шляхом створення резервних копій;

– виконання спеціальних вимог для використовуваних імпортних апаратних засобів;

– захист від витоку по побічних каналах технічних засобів, призначених для обробки й зберігання конфіденційної інформації;

– захист баз даних різного рівня;

– захист каналів передачі інформації;

– підтвердження авторства повідомлень із використанням електронного цифрового підпису інформації.

При формуванні на основі первинних даних аналітичної й агрегованної інформації у відповідних інформаційно-аналітичних системах повинні бути використані засоби криптографічного захисту, що відповідають категорійності формованої інформації.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

SolarWinds Security Event Manager

Найкраще для малого та великого бізнесу.

Ціна: забезпечує повнофункціональну пробну версію протягом 14 днів. Ціна на виріб починається від $4500.

SolarWinds Security Event Manager – це система виявлення вторгнень у мережу та хост. Він здійснює моніторинг у реальному часі, реагує та повідомляє про загрози безпеці. Він має високоіндексовані можливості пошуку журналів. Це хмарне масштабоване рішення.

Особливості:

– Дані про загрози постійно оновлюватимуться.

– Він має функції для інформації про безпеку та керування подіями.

– Він пропонує функції кореляції журналу та архіву подій журналу.

– Він надає повний набір інтегрованих інструментів звітності.

Вердикт: Solarwinds Security Event Manager – це хмарне рішення, розроблене для постачальників керованих послуг як комплексне рішення інструменту SIEM.

SecPod SanerNow

Найкраще для малого та великого бізнесу.

Платформа кібергігієни SanerNow надає розширене рішення для керування вразливістю для забезпечення безперервного ризику безпеки та відповідності вимогам для запобігання кібератакам. Це вдосконалена платформа

керування вразливістю, яка об’єднує оцінку вразливості з миттєвим виправленням у єдину уніфіковану консоль.

Він шукає вразливості, неправильні конфігурації тощо та надає засоби керування й методи для миттєвого й автоматичного виправлення.

Завдяки власно створеній системі кожен крок керування вразливістю, від сканування до виправлення, можна автоматизувати. SanerNow допомагає вам посилити безпеку вашої організації та запобігти кібератакам.

Особливості:

– Він використовує інтелектуальний і легкий багатофункціональний агент, який виконує всі завдання.

– Оцінюючи потенціал ризику, високоточні атаки тощо, SanerNow ефективно визначає пріоритети вразливостей для легкого усунення.

– Завдяки інтегрованим виправленням ви можете швидко усунути вразливості в IT-активах.

– За допомогою засобів керування виправленням, окрім виправлення, пом’якшення ризиків безпеці стає легшим.

– З єдиної хмарної консолі ваша організація може ефективно пом’якшувати вразливості тощо.

– За допомогою SanerNow ви можете керувати вразливістю в реальному часі, від сканування до виправлення.

Вердикт: із SanerNow ви отримуєте повне рішення для кібербезпеки, яке підніме ваш процес керування вразливістю на наступний рівень, керуючи іншими ризиками безпеки з тієї самої консолі. Крім того, він може замінити кілька рішень, які ви використовуєте для керування вразливістю та керування виправленнями, допомагаючи вам ефективніше керувати поверхнями атак.

Intruder

Найкраще для малого та великого бізнесу. Ціна: доступна 30-денна безкоштовна пробна версія. Він включає три тарифні плани, тобто Essential, Pro та Verified. Зв'яжіться з ними, щоб отримати докладнішу інформацію про ціни.

Intruder – це найпопулярніший хмарний мережевий сканер уразливостей, який допомагає вам знаходити слабкі місця кібербезпеки у найбільш уразливих системах, щоб уникнути дорогого витоку даних. Це правильне рішення для ваших проблем кібербезпеки. Це значною мірою допомагає заощадити ваш час.

Особливості:

– Понад 9000 вразливостей системи безпеки.

– Необмежена кількість сканувань на вимогу.

– Необмежена кількість облікових записів користувачів.

– Перевіряє наявність недоліків веб-додатків, таких як впровадження SQL і міжсайтовий сценарій.

– Сповіщення про нові загрози.

– Smart Recon

– Перегляд мережі

– Доступні скани PCI ASV.

Вердикт: Intruder – це універсальне рішення для всіх ваших потреб у сфері кібербезпеки.

BitLocker та TrueCrypt

Microsoft поставляє систему шифрування BitLocker в операційних системах Windows 10/11, але вона доступна тільки в high-end версіях Enterprise і Ultimate. У той же час існує потужна альтернатива BitLocker: програма TrueCrypt з відкритим вихідним кодом і із кращою гнучкістю. У рамках виконання магістерського проектування вирішили зрівняти функції й продуктивність обох рішень.

TrueCrypt дозволяє шифрувати й захищати паролем вашу систему цілком "на льоту", при цьому утиліта незначно позначається на продуктивності й на часі автономної роботи.

Вважається, що навряд чи має сенс обговорювати причини, по яких багато користувачів переходять на шифрування даних або системи цілком. Втрата даних – це одна проблема, але її можна передбачити (нехай навіть іноді це досить накладно), але якщо дані потраплять у руки зловмисника, то це може являти серйозну загрозу для бізнесу або для безпеки.

Цього разу вирішили довідатися, чи вірні дані результати тільки для TrueCrypt, або впровадження Microsoft BitLocker дає таку ж перевагу – ця система шифрування поставляється з версіями Enterprise і Ultimate операційних систем Windows 10/11. Чи має зміст доплачувати за версії Windows, що підтримують BitLocker. Або можна використовувати утиліту TrueCrypt, що зробить те ж саме, але зовсім безкоштовно.

Ще однією причиною для тестів рішень шифрування можна вважати появу додаткових нових інструкцій AES (AES-NI) у двоядерних процесорах Intel Core i5 для масового ринку (Clarkdale). Чи можуть функції шифрування BitLocker і TrueCrypt вигравати від цих інструкцій.

BitLocker на Windows 10/11

Функція BitLocker є цілком логічним рішенням для шифрування томів, при цьому можливо використовувати цю функцію для різних томів, які як містять у собі кілька твердих дисків, так і займають тільки частину простору вінчестера. BitLocker для своєї роботи вимагає двох розділів NTFS. Один з розділів – завантажувальний, котрий не можна шифрувати; другий розділ призначений для вашої операційної системи.

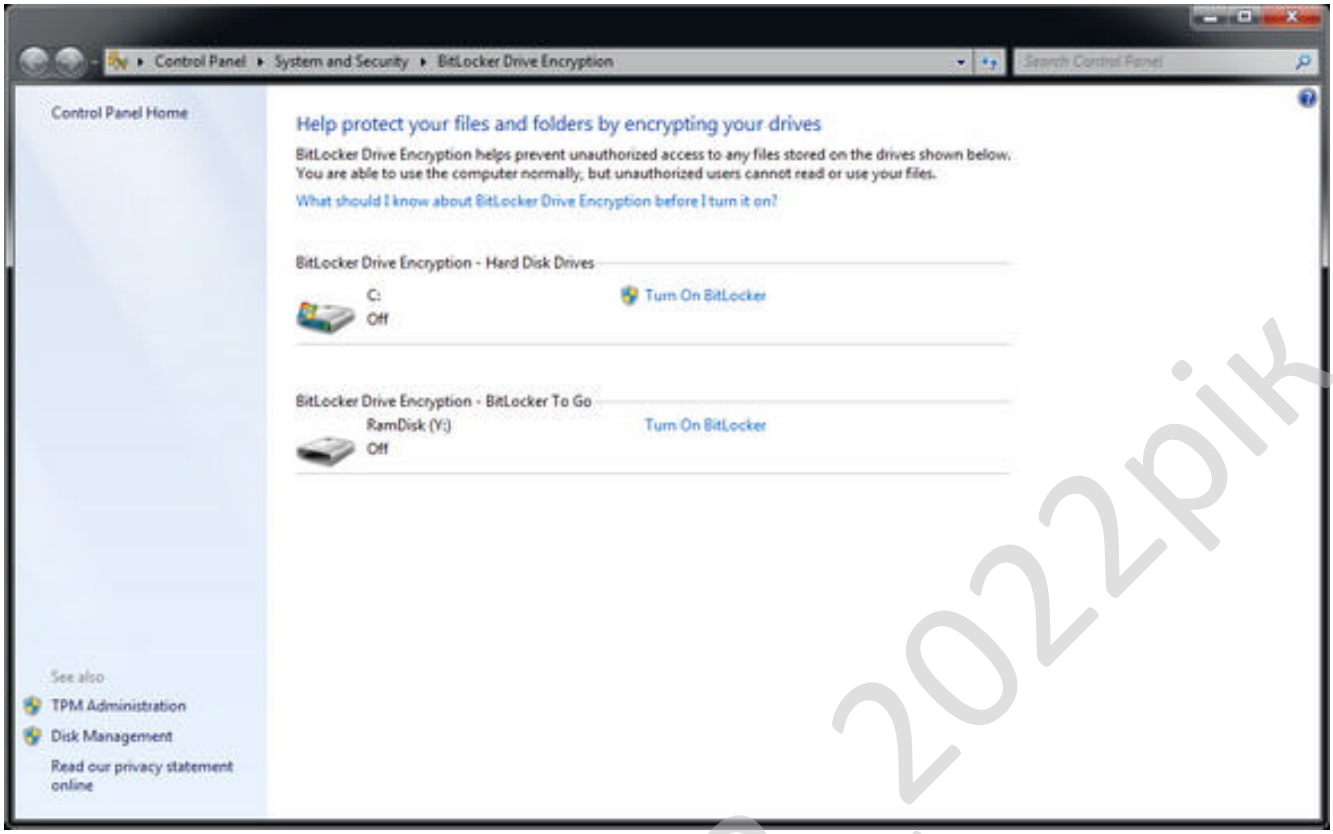

Рисунок 2.1 – Вивід панелі керування BitLocker

Дане рішення використовує 128-бітне шифрування AES, при цьому доступні різні способи автентифікації. Microsoft підтримує автентифікацію через модуль Trusted Platform Module (TPM), TPM з PIN-кодом, TPM із ключем на USB-брелоці й комбінацію TPM, PIN-коду й ключа на USB-брелоці. Залежно від функціональності кожного рішення, можливо вибирати різні опції. Наприклад, проста підтримка модуля TPM приведе до втрати працездатності системи, якщо ви встановите жорсткий диск на інший комп'ютер, а інші рішення вимагають активної автентифікації.

Можливо вивести панель керування BitLocker, увівши відповідне слово в панелі пошуку меню програм або ручним запуском відповідного ярлика.

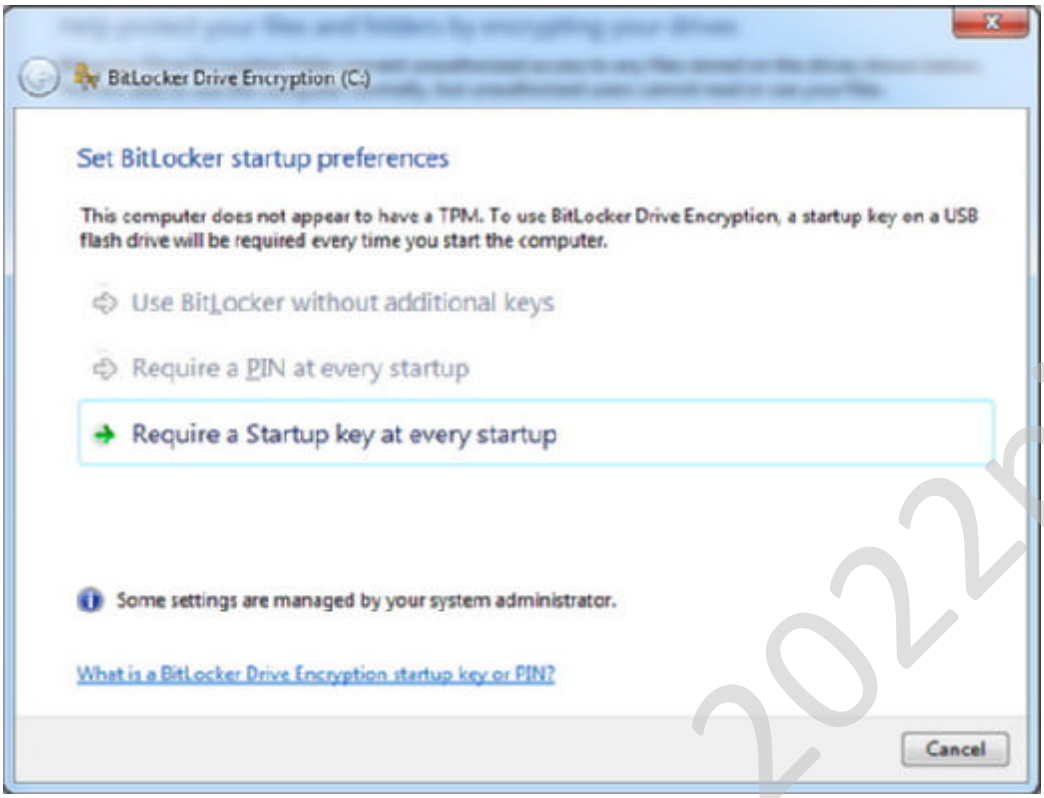

Рисунок 2.2 – Запуск панелі керування BitLocker

Використання BitLocker без додаткових заходів було неможливо, оскільки в тестової системи не було модуля TPM. Тому обрано опцію перевірки ключа запуску при кожному завантаженні системи. Ключ, як правило, перебуває на USB-брелоці.

Ви також можете вибрати, куди копіювати ключ відновлення, що буде потрібно, якщо ви забудете PIN-код або втратите USB-брелок, що містить ключ.

Можна приступати до шифрування.

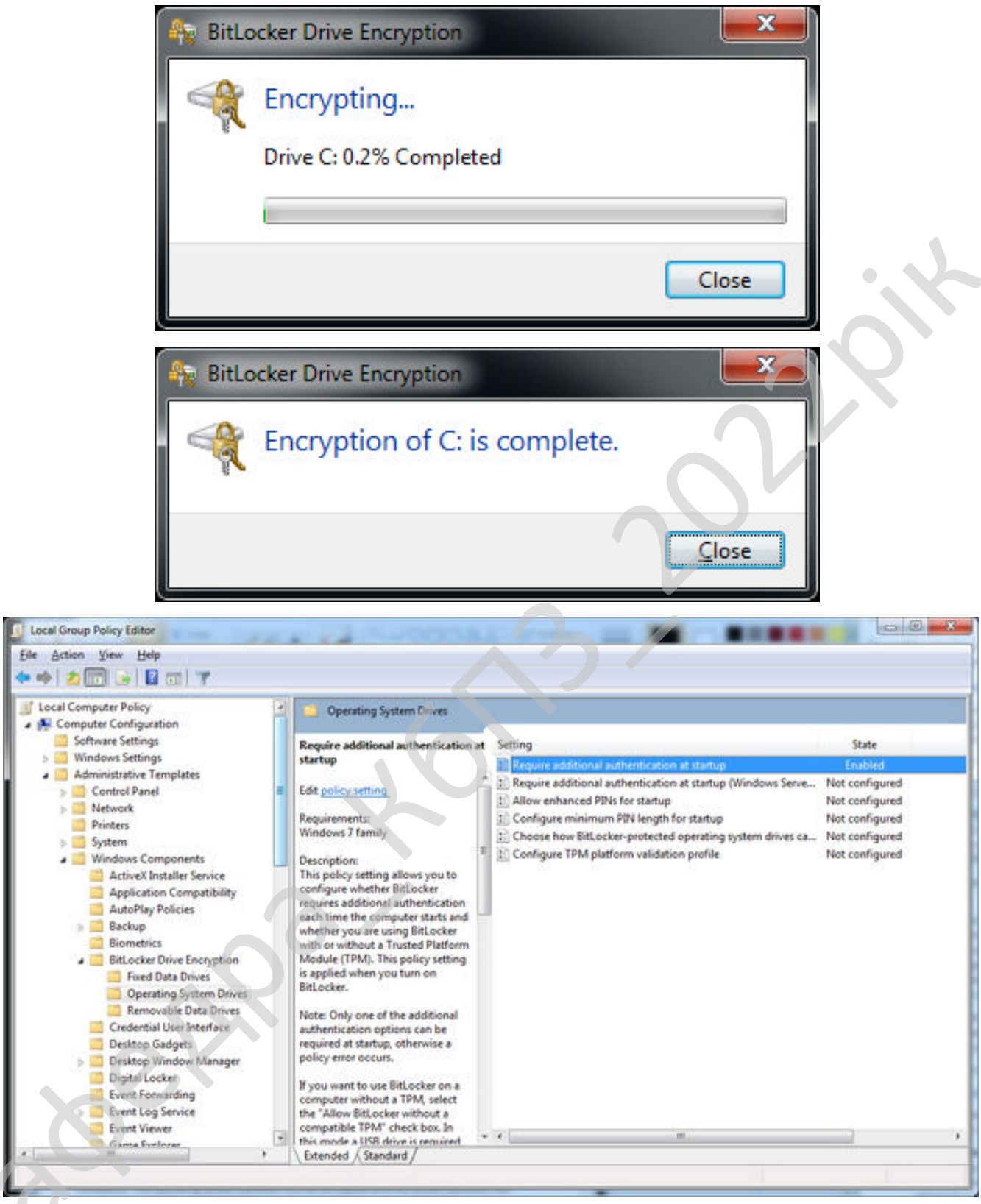

Рисунок 2.3 – Виконання операції шифрування BitLocker

Таблиця 2.1 – Тестова конфігурація

Системне апаратне забезпечення	
Материнська плата (Socket LGA1156)	Gigabyte P55 A-UD7 (Rev. 1.0), чипсет: P55, BIOS: F3 (01/26/2010)
CPU Intel	Intel Core i 5-750 (45 нм, 2,66 ГГц, 4x 256 кбайт кеша L2 і 8 Мбайт кеша L3, TDP 95 Вт)
Пам'ять DDR3 (два канали)	2x 2 Гбайт DDR 3-1600 (OCZ OCZ3G2000LV4GK), DDR 3-1333 8-8-8-24 1T
Жорсткий диск	Western Digital VelociRaptor, 300 Гбайт (WD3000HLFS), 10000 об/хв, SATA/300, кеш 16 Мбайт
Відеокарта	Sapphire Radeon HD 5850, GPU: Cypress (725 МГц), пам'ять: 1024 Мбайт GDDR5 (2000 МГц), число потокових процесорів: 1440
Блок живлення	PC Power & Cooling, Silencer 750EPS12V 750W
Системне ПЗ і драйвери	
Операційна система	Windows 10/11
Драйвери й налаштування	
Драйвери чипсета Intel	Chipset Installation Utility Ver. 9.1.1.1025
Графічні драйвери ATI	Radeon Version 10.1

Для тестів ми використовували материнську плату Gigabyte P55 A-UD7, процесор Core i 5-750 і відеокарту Sapphire Radeon HD 5850.

Таблиця 2.2 – Тести й налаштування

Тести додатків	
7-zip	Version 9.1 beta LZMA2 Syntax "a -t7z -r -m0=LZMA2 -mx=5" Benchmark: 2010-THG-Workload
PCMark Vantage	Version 1.0.1.0
Sysmark Preview 2007	Version 1.06
WinRAR	Version 3.92 Beta 1 RAR Syntax "a -r -m3" Benchmark: 2010-THG-Workload
WinZip 14	Version 14.0 Pro (8652) WinZIP Commandline Version 3 ZIPX Syntax "-a -ez -p -r" Benchmark: 2010-THG-Workload

Результати тестів

Архіватори

Було запущено кілька утиліт стиску файлів, щоб подивитися, чи зробить який-небудь вплив використання зашифрованих розділів, однак обчислювальне навантаження стиску виявилося набагато більше істотної, чим навантаження на шифрування/розшифровку даних. Процес архівації виявився ледве повільніше на зашифрованих розділах, але різниця дуже мала.

PCMark Vantage

Отримано найвищий результат продуктивності без шифрування системного розділу, але технологія BitLocker, як видно, виграє від нових інструкцій Intel AES (AES-NI) у випадку двоядерного Core i 5-661, оскільки

результати продуктивності виявилися такими ж високими, як і у випадку незашифрованого розділу.

Тест Memories дав більше високі результати на чотирьохядерному Core i 5-750; BitLocker на двоядерному процесорі дає не такий відчутний провал, що можна, знову ж, зв'язати з апаратною підтримкою прискорення AES на цьому CPU.

SYSmark Preview

Тест 3D-моделювання SYSmark виконується швидше на незашифрованому розділі.

Тест створення відео використовує Sony Vegas і Adobe After Effects, при цьому ми спостерігаємо перевагу чотирьохядерного процесора Core i 5-750, але ми бачимо непогані результати й на двох ядрах – зокрема, через апаратну підтримку шифрування AES у випадку BitLocker. Але на розділі TrueCrypt результати в цьому тесті все-таки не такі високі.

Спостерігається невелика різниця в тесті продуктивності. Знову ж, незашифрований системний розділ дає кращу продуктивність, але різниця невелика.

Висновок

Два ядра, чотири ядра, середні тактові частоти, високі тактові частоти, підтримка AES: всі ці фактори впливають не так сильно. Робота шифрування в реальному часі на системному жорсткому диску впливає на продуктивність, будь то BitLocker від Microsoft або TrueCrypt.3а. Однак з падінням продуктивності зашифрованих систем цілком можна упокоритися, та й воно виявляється практично однаковим незалежно від того, тестуємо двоядерний Core i 5-600 або чотирьохядерний Core i 5-700. Втім, все-таки рекомендуємо бути дуже обережним зі старими й, особливо, одноядерними системами, де шифрування в реальному часі може більш істотно знижувати продуктивність.

В Microsoft BitLocker все-таки є деякі переваги завдяки новим інструкціям Intel AES. Вони присутні на всіх двоядерних настільних процесорах Core i5 (і на

більшості мобільних моделей), та й також будуть інтегровані на всіх прийдешніх процесорах Intel для масового ринку або для більше високого сегмента. Але, знову ж: переваги не такі істотні, так що не варто ухвалювати рішення щодо покупки тільки на основі цього фактора.

Отже, вплив на продуктивність незначний, тому ми одержуємо, в основному, битву різних наборів функцій. І цю битву Microsoft уже не може виграти з поточною версією BitLocker. І справа не в тому, що дана функція потенційно менш потужна, просто вона націлена на корпоративний ринок. Шифрування BitLocker підтримує TPM і множинні способи автентифікації, але гнучкість у реальному житті досить істотно обмежена.

Утиліта TrueCrypt набагато більш гнучка завдяки підтримці декількох операційних систем, можливості вибору алгоритмів шифрування й декількох способів захисту системи. Ви навіть можете створювати сховані розділи, а також захищати системи на основі схованих основних і облудного вторинного розділів, які додають ще один психологічний рівень захисту – атакуючі навіть не знають, що вони мають справу не з основною операційною системою. Нарешті, нам сподобався той факт, що утиліта TrueCrypt прекрасно працює на будь-якій системі, тобто вона дійсно апаратно незалежна. Оскільки розділ буде легко перетворити назад у незашифрований, цю утиліту можна без проблем спробувати в роботі.

2.2 Обгрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в

чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи забезпечення конфіденційності даних хмарних сервісів.

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

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі.

Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Дамо опис алгоритму AES. У його основі лежить алгоритм Rijndael

Формат блоків даних і число раундів

RIJNDAEL – це симетричний блоковий шифр, що оперує блоками даних розміром 128 і довжиною ключа 128, 192 або 256 біт – назва стандарту відповідно "AES-128", "AES-192", і "AES-256".

Уведемо наступні позначення:

– Nb – число 32-бітних слів які містяться у вхідному блоці, Nb – 4;

– Nk – число 32-бітних слів, що втримуються в ключі шифрування, Nk = 4,6,8;

– Nr – число раундів шифрування, як функція від Nb і Nk, Nr = 10,12,14.

Вхідні (input), проміжні (state) і вихідні (output) результати перетворень, виконуваних у рамках криптоалгоритму, називаються станами (State). Стан можна представити у вигляді матриці 4 x Nb, елементами якої є байти (чотири рядки по Nb байт) у порядку S00, S10, S20, S30, Sm, Sn, S21, S31. Рисунок 3.1 демонструє таке подання, що носить назву архітектури «Квадрат».

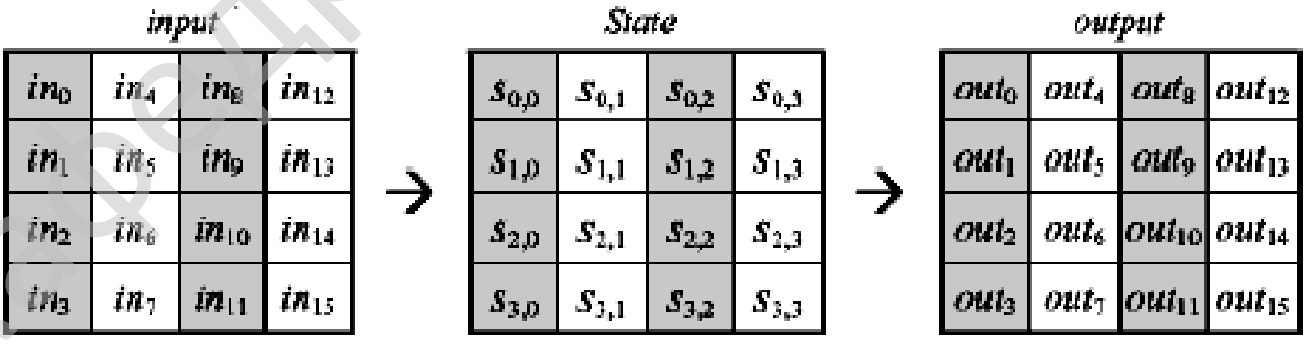

Рисунок 3.1 – Приклад подання блоку у вигляді матриці 4xNb

На старті процесів шифрування й дешифрування масив вхідних даних те, input перетвориться в масив State за правилом:

$$s[r,c] = in[r + 4c], \quad (3.1)$$

де $0 < r < 4$ і $0 < c < Nb$.

Наприкінці дії алгоритму виконується зворотне перетворення:

$$out[r + 4c] = s[r,c], \quad (3.2)$$

де $0 < r < 4$ і $0 < c < Nb$ – вихідні дані виходять із байтів стану в тому же порядку.

Чотири байти в кожному стовпці стану являють собою 32-бітне слово, якщо r – номер рядка в стані, те одночасно він є індексом кожного байта в цьому слові. Отже стан можна представити як одномірний масив 32-бітних слів $w_0,...,w_n$, де номер стовпця стану с є індекс у цьому масиві. Тоді стан можна представити так:

$w_0 = s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0}$	$w_2 = s_{0,2}, s_{1,2}, s_{2,2}, s_{3,2}$
$w_1 = s_{0,1}, s_{1,1}, s_{2,1}, s_{3,1}$	$w_3 = s_{0,3}, s_{1,3}, s_{2,3}, s_{3,3}$

Ключ шифрування також як і масив State представляється у вигляді прямокутного масиву із чотирма рядками. Число стовпців цього масиву дорівнює Nk.

Для алгоритму AES число раундів Nr визначається на старті залежно від значення Nk (таблиця 3.1):

Таблиця 3.1 – Залежність значення Nr від Nk і Nb

	Nk	Nb	Nr
AES– 128	4	4	10
AES– 192	6	4	12
AES – 256	8	4	14

Функція зашифрування

Введемо наступні позначення:

– SubBytes() – заміна байтів – побайтова нелінійна підстановка в State-блоках (S-Box) з використанням фіксованої таблиці замін розмірністю 8x256

(affain map table).

– ShiftRows() – зрушення рядків – циклічне зрушення рядків масиву State на різну кількість байт.

– MixColumns() – перемішування стовпців – множення стовпців стану, розглянутих як багаточлени над GF(2^8).

– AddRoundKey() – додавання з раундовим ключем – поразрядне XOR умісту State з поточним фрагментом розгорнутого ключа.

На псевдокоді операція зашифрування виглядає в такий спосіб:

```
Cipher[byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Після заповнення масиву State елементами вхідних даних до нього застосовується перетворення AddRoundKey, далі, залежно від величини Nk, масив State піддається раундовій трансформації 10, 12 або 14 разів, причому у фінальний раунд є трохи вкороченим – у ньому відсутнє перетворення MixColumns. Вихідними даними описаної послідовності операцій є шифротекст – результат дії функції зашифрування AES.

Функції розшифрування

У специфікації алгоритму AES пропонуються два види реалізацій функції розшифрування відмінних друг від друга послідовністю додатка перетворень зворотних перетворенням функції зашифрування й послідовністю планування ключів.

Введемо наступні позначення:

– InvSubBytes() – зворотна SubBytes() заміна байтів – побайтова нелінійна підстановка в S-блоках з використанням фіксованої таблиці замін розмірністю 8x256 (inverse affain map).

– InvShiftRows() – зворотне зрушення рядків ShiftRows() – циклічне зрушення рядків масиву State на різну кількість байт.

– InvMixColumns() – відновлення значень стовпців – множення стовпців стану, розглянутих як багаточлени над GF(2^8).

Функція зворотного розшифрування

Якщо замість SubBytes(), ShiftRows(), MixColumns() і AddRoundKey() у зворотній послідовності виконати інверсні їм перетворення, можна побудувати функцію зворотного розшифрування. При цьому порядок використання раундових ключів є зворотним стосовно тому, що використовується при зашифруванні. На псевдокоді вона виглядає так:

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1
      InvShiftRows(state)
      InvSubBytes(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)
   end for

   InvShiftRows(state)
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

Алгоритм зворотного розшифрування, описаний вище має порядок додатка операцій – функцій зворотний порядку операцій в алгоритмі прямого зашифрування, але використовує ті ж параметри розгорнутого ключа.

Функція прямого розшифрування

Змінивши певним чином послідовність планування ключа можна побудувати ще один алгоритм – алгоритм прямого розшифрування.

```
EqInvCipher(byte in[4*Nb], byte out[4*Nb], word dw[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, dw[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1
      InvSubBytes(state)
      InvShiftRows(state)
      InvMixColumns(state)
      AddRoundKey(state, dw[round*Nb, (round+1)*Nb-1])
   end for

   InvSubBytes(state)
   InvShiftRows(state)
   AddRoundKey(state, dw[0, Nb-1])

   out = state
end
```

Дві наступні властивості дозволяють зробити це:

– Порядок додатка функцій SubBytes() і ShiftRows() не грає ролі. Те ж саме вірно й для операцій InvSubBytes() і InvShiftRows(). Це відбувається тому, що функції SubBytes() і InvSubBytes() працюють із байтами, а операції ShiftRows() і InvShiftRows() зрушують цілі байти, не зачіпаючи їхніх значень.

– Операція MixColumns() є лінійною щодо вхідних даних, що означає InvMixColumns(State XOR RoundKey) = InvMixColumns(State) XOR InvMixColumns(RoundKey).

Ці властивості функцій алгоритму шифрування дозволяють змінити порядок застосування функцій InvSubBytes() і InvShiftRows(). Функції AddRounKey() і InvMixColumns() також можуть бути застосовані у зворотному порядку, але за умови, що стовпці (32-бітні слова) розгорнутого ключа розшифрування попередньо пропущені через функцію InvMixColumns().

Таким чином, можна реалізувати більше ефективний спосіб розшифрування з тим же порядком додатка функцій як і в алгоритмі зашифрування.

Алгоритм вироблення ключів (Key Schedule)

Введемо наступні позначення:

– Rcon[i] – масив 32-бітних раундових констант;

– RotWord() – операція циклічної перестановки вхідного 4-х байтного слова у вихідне за наступним правилом [a0, ai, a2, a3] —> [ax, a2, a3, a0];

– SubWord() – операція заміни в 4-х байтному слові за допомогою S-Box кожного байта;

⊕ – операція що виключає або – XOR.

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
   word  temp

   i = 0

   while (i < Nk)
      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
      i = i+1
   end while

   i = Nk

   while (i < Nb * (Nr+1)]
      temp = w[i-1]
      if (i mod Nk = 0)
         temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
      else if (Nk > 6 and i mod Nk = 4)
         temp = SubWord(temp)
      end if
      w[i] = w[i-Nk] xor temp
      i = i + 1
   end while
end
```

Раундові ключі виходять із ключа шифрування за допомогою алгоритму вироблення ключів. Він містить два компоненти: розширення ключа (Key Expansion) і вибір раундового ключа (Round Key Selection). Основні принципи алгоритму виглядають у такий спосіб:

– загальне число бітів раундових ключів дорівнює довжині блоку, помноженої на число раундів, плюс 1;

– ключ шифрування розширюється в розширений ключ (Expanded Key);

– раундові ключі беруться з розширеного ключа в такий спосіб: перший раундовий ключ містить перші Nb слів, другий – наступні Nb слів і т.д.

Розширення (планування) ключа

Розширений ключ являє собою лінійний масив w[i] складається з Nb*(Nr + 1) 4-х байтових слів, i = Nb*(Nr + 1).

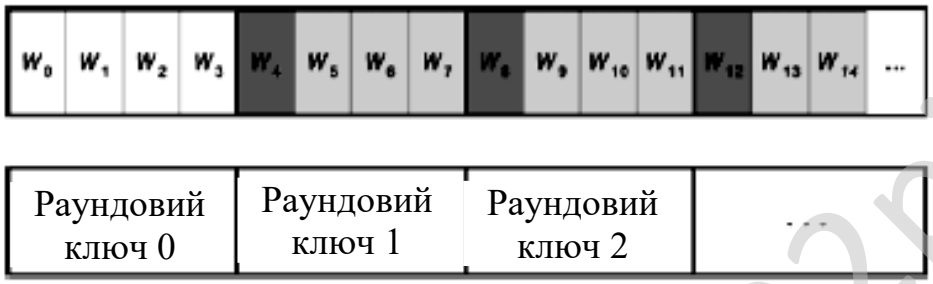

Рисунок 3.2 – Процедури розширення й виборки раундового ключа для Nk = 4

Ясно-сірим кольором виділені слова розширеного ключа, які формуються без використання функцій SubWord() і RotWord().

Темно-сірим кольором, виділені слова розширеного ключа, при обчисленні яких використовуються перетворення SubWord() і RotWord().

Перші Nk слів містять ключ шифрування. Кожне наступне слово w[i] виходить за допомогою XOR попереднього слова w[i-1] і слова на Nk позицій раніше:

$$W[i\text{-}Nk]: w[i]= w[i\text{-}1] \oplus w[i\text{-}Nk]. \quad (3.3)$$

Для слів, позиція яких кратна Nk, перед XOR застосовується перетворення до w[i-1], а потім ще додається раундова константа Rcon[i]. Перетворення реалізується за допомогою двох додаткових функцій: RotWord() і SubWord().

Значення Rcon[j] дорівнює 2j-1. Значення w[i] у цьому випадку визначається вираженням:

$$w[i] = SubWord(RotWord(w[i\text{-}1]) \oplus Rcon[i/Nk] \oplus M[i\text{-}Nk]. \quad (3.4)$$

Вибір раундового ключа і-тий раундовий ключ вибирається зі слів масиву розширеного ключа в проміжку від W[Nb * i] до W[Nb * (i+1)].

Для функції зашифрування розширений ключ генерується алгоритмом. Для функції зворотного розшифрування використовується цей же ключ, але у зворотній послідовності починаючи з останнього раундового підключа зашифрування.

Для функції прямого розшифрування використовується трохи модифікований алгоритм планування ключа. При формуванні розгорнутого ключа в процедуру планування необхідно додати наприкінці додаткове перетворення, причому розширений ключ використовується при прямому розшифруванні в тій же послідовності, що й при зашифруванні.

Додаткове перетворення розширеного ключа для функції прямого розшифрування:

```
for i = 0 step 1 to (Nr+1)*Nb-1
   dw[i] = w[i]
end for

for round = 1 step 1 to Nr-1
   InvMixColumns(dw[round*Nb, (round+1)*Nb-1])
end for
```

Раундове перетворення

Раундове перетворення складається з послідовного застосування до масиву State ряду трансформацій.. Зараз обговоримо деталі його реалізації.

Нелінійна заміна байтів масиву стану за допомогою трансформації SubBytes має вигляд:

$$\lambda(f) = ((X^4 + X^3 + X^2 + X + 1) \cdot f + X^6 + X^5 + X + 1) \bmod (X^8 + 1) \qquad (3.5)$$

Багаторазове обчислення в процесі зашифрування даного вираження робило б невиправдане обчислювальне навантаження на виконуючу систему, тому для практичної реалізації найбільш прийнятним рішенням є використання попереднє обчисленої таблиці заміни S-Box. Логіка роботи S-Box при перетворенні байта (xy} відбита в шестнадцятковому виді на рисунку 3.3:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рисунок 3.3 – Таблиця S-Box заміни байт

Її використання зводить операцію SubBytes до найпростішої вибірки байта з масиву λ (f) = Sbox[f].

У функціях розшифрування застосовується операція зворотна InvSubBytes().

Вона реалізується так само просто, як і попередня за допомогою інверсної таблиці S-Box – λ^{-1}(f) = InvSbox[f]. Її логіка роботи при перетворенні байта (xy} відбита в шестнадцятковому виді на рисунку 3.4.

Рисунок 3.5 ілюструє застосування перетворення заміни байт до стану у функціях зашифрування й розшифрування.

У перетворенні зрушення рядків (ShiftRows) останні 3 рядка стану циклічно зрушуються ВЛІВО на різне число байтів. Рядок 1 зрушується на C1 байт, рядок 2 – на C2 байт, і рядок 3 – на C3 байт. Значення зрушень C1, C2 і C3 в Rijndael залежать від довжини блоку Nb .

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рисунок 3.4 –Таблиця S-Box інверсної заміни байт

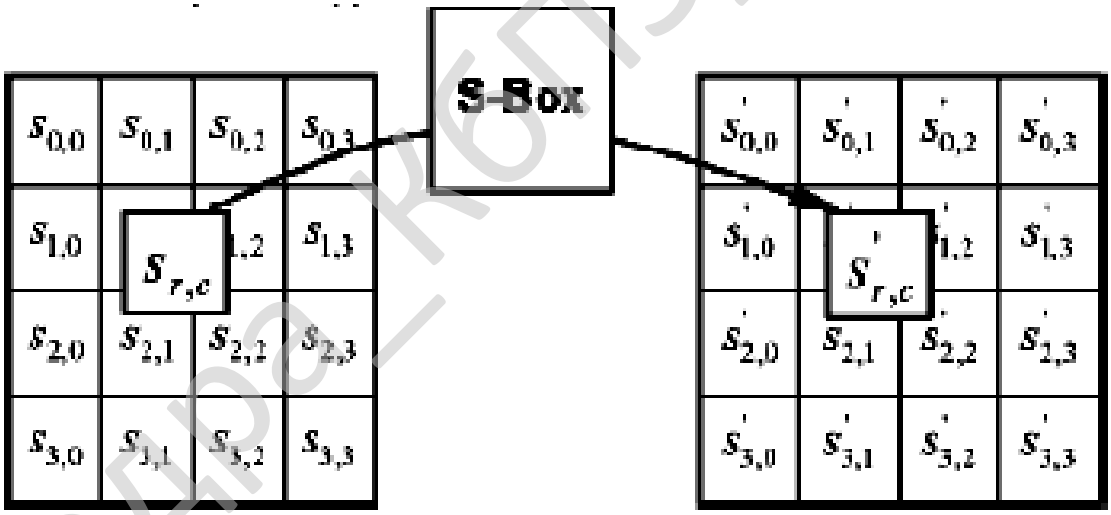

Рисунок 3.5 – Перетворення стану за допомогою таблиці заміни S-Box

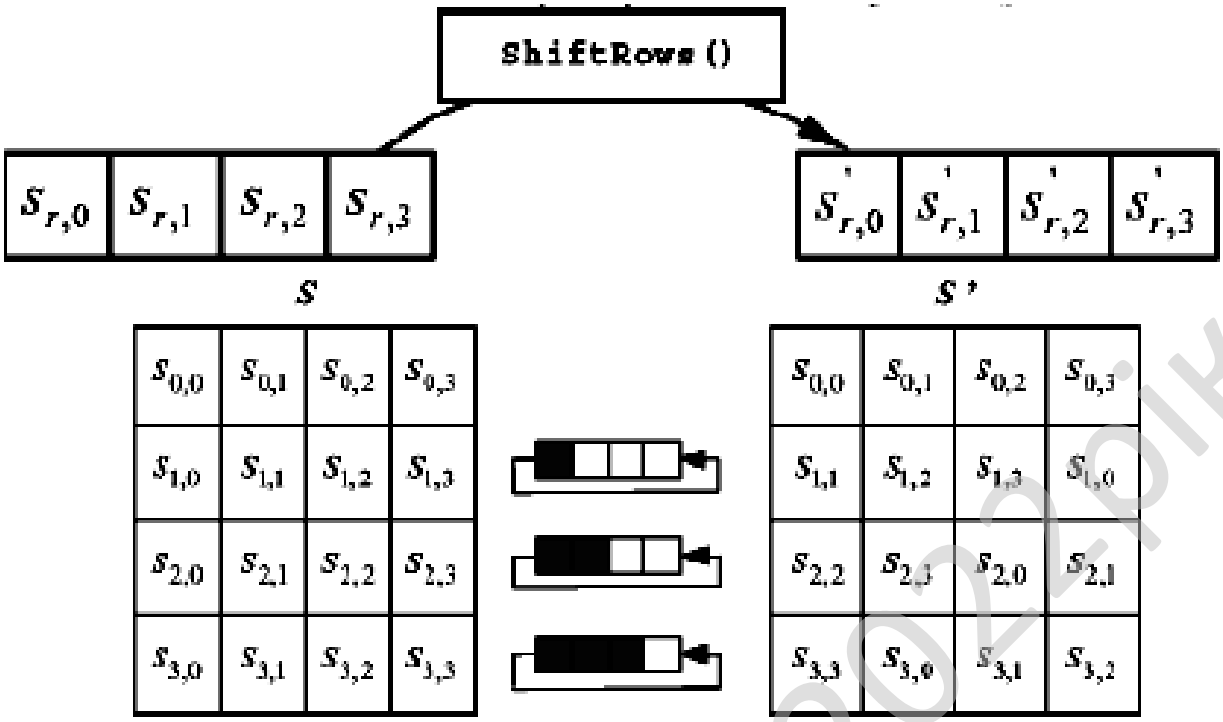

Рисунок 3.6 – Перетворення зрушення рядків у функції зашифрування

У перетворенні зворотного зрушення рядків InvShiftRows останні 3 рядка стану циклічно зрушуються ВПРАВО на різне число байтів. Рядок 1 зрушується на C1 байт, рядок 2 – на C2 байт, і рядок 3 – на C3 байт.

Перемішування стовпців

У перетворенні перемішування стовпців (MixColumns) стовпці стану розглядаються як багаточлени над GF(28) і піддаються перетворенню c(x) = c * gmod(Y^4 +1), де c = (X,1,1,X+1), тобто множаться за модулем $x^4 + 1$ на багаточлен c(x), що виглядає, як: $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

Застосування цієї операції до всім чотирьох стовпцям стану позначається, як MixColumns(State). Рисунок 3.7 демонструє застосування перетворення MixColumns до стовпця стану.

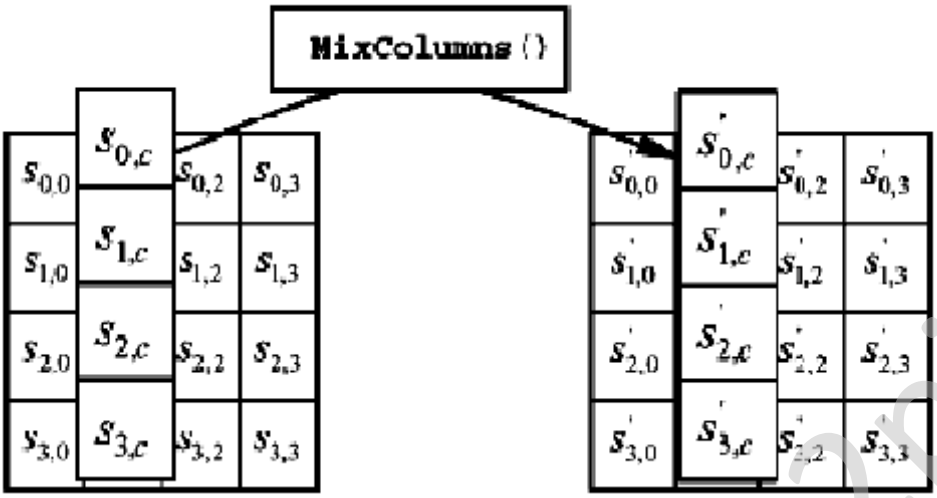

Рисунок 3.7 – Операція перемішування діє на стовпці стану

У зворотному перетворенні InvMixColumns стовпці стану розглядаються як багаточлени над GF(2^8), але, природно, піддаються зворотному перетворенню.

Додавання раундового ключа AddRoundKey()

У даній операції раундовий ключ додається до стану за допомогою поразрядного XOR. Довжина ключа (в 32-розрядних словах) дорівнює довжині блоку Nb . Рисунок 3.8 ілюструє дію даної операції на стан. Це перетворення позначається як AddRoundKey(State, RoundKey).

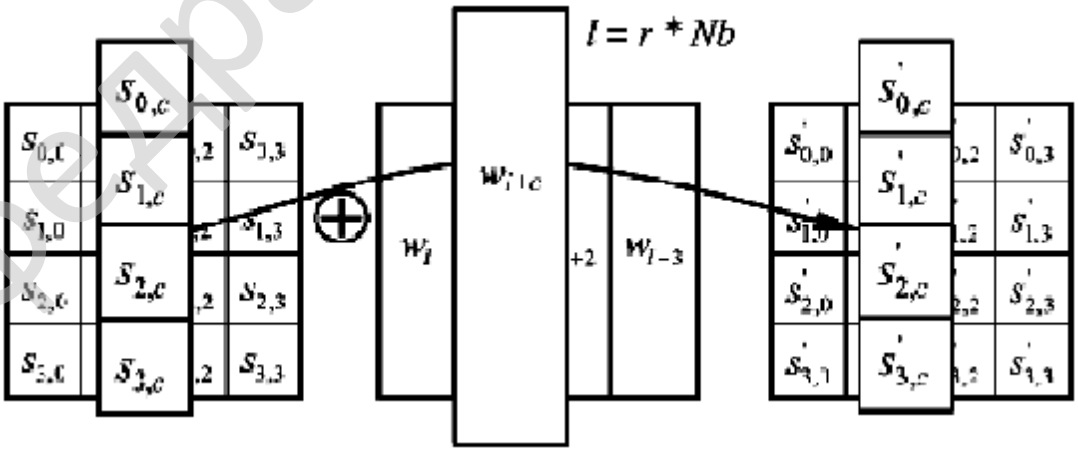

Рисунок 3.8 – AddRoundKey

Основні особливості AES

У висновку сформулюємо основні особливості AES:

– нова архітектура «Квадрат», що забезпечує швидке розсіювання й перемішування інформації, при цьому за один раунд перетворенню піддається весь вхідний блок;

– байт-орієнтована структура, зручна для реалізація на 8– розрядні мікро контролерах;

– всі раундові перетворення суть операції в кінцевих полях, що допускають ефективну апаратну й програмну реалізацію на різних платформах.

3.2 Розробка структурної схеми

На рисунку 3.9 зображена структурна схема системи забезпечення конфіденційності даних хмарних сервісів.

Забезпечення безпеки інформації при зберіганні й обробці більших інформаційних масивів у хмарних сервісах – 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.

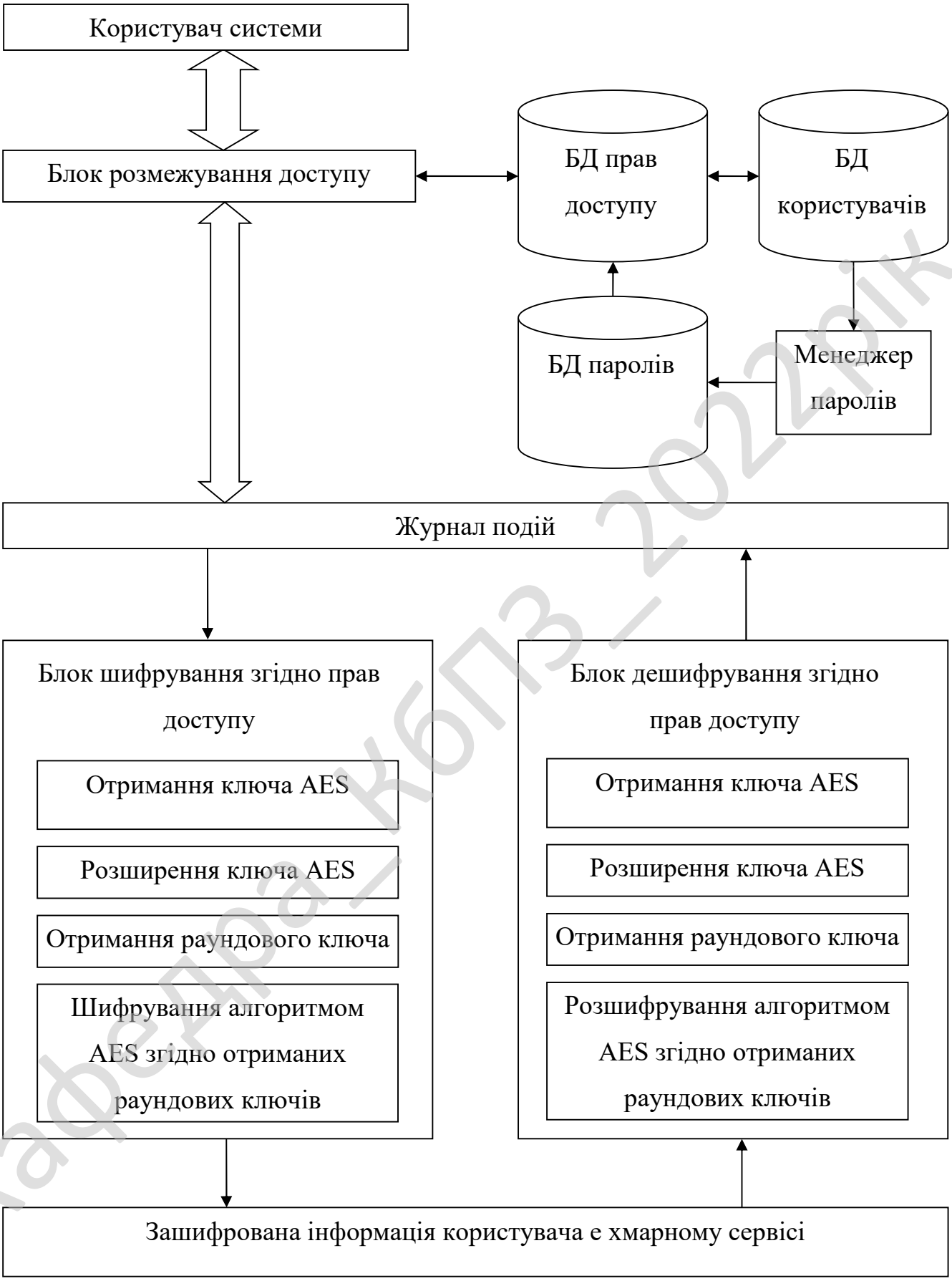

Рисунок 3.9 – Структурна схема системи

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. Тому що 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.

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.

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 трафіка між вузлами, що, як відомо, вирішується шляхом суцільного заповнення каналу псевдовипадковими послідовностями.

Виходячи зі структурної схеми системи зображеної на рисунку 3.9, система забезпечення конфіденційності даних хмарних сервісів, працює наступним чином.

Спершу при вході в систему, користувач звертається до блоку розмежування доступу.

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 зберігання інформації.

Після підтвердження прав доступу, та правильності введеного паролю, користувачеві видається сеансовий ключ AES для роботи з інформацією.

У блоці шифрування згідно прав доступу, з отриманого ключа AES відбувається його розширення, та обирається ключ ітерації, за допомогою яких й відбувається шифрування інформації алгоритмом AES.

Процедура дешифрування відбувається аналогічним чином.

3.3 Розробка функціональної схеми

На рисунку 3.10 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми.

– Блок розмежування доступу.

– Блок менеджеру паролів.

– Блок журналювання подій.

– Допомога.

– Блоки шифрування та дешифрування інформації згідно алгоритму AES.

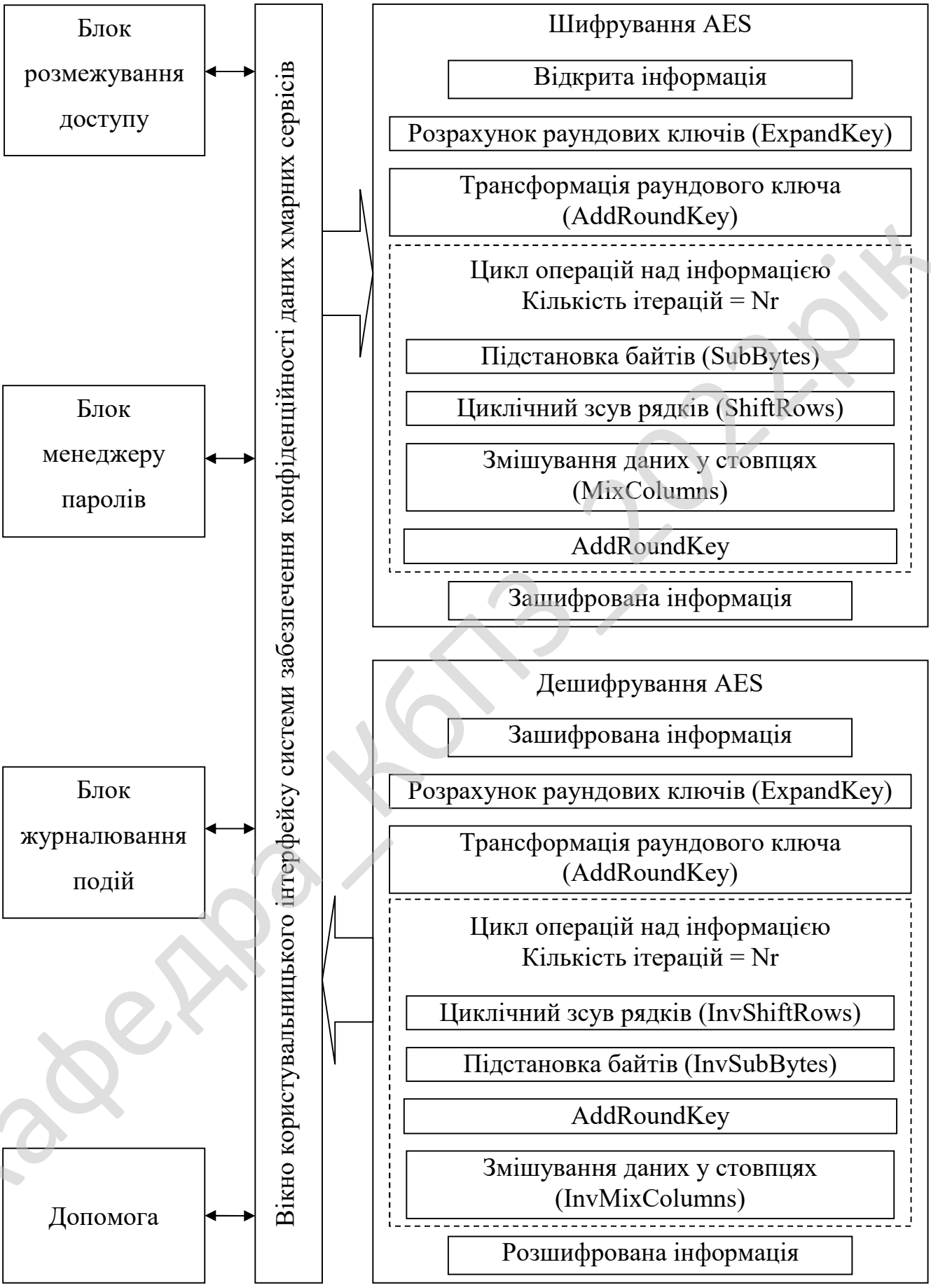

Рисунок 3.10 – Функціональна схема системи

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. 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.

Блок розмежування доступу. Призначений для організації безпечного доступу співтовариства користувачів до захищених ресурсів. Члени цього співтовариства, використовуючи програму, одержують визначені переваги. Це дає наступні можливості:

– Надавати користувачам доступ до інформації (наприклад, групи структурних схем, адресні довідники відділів або пошук співробітників) і ресурсам (наприклад, устаткування або облікові записи у внутрішніх системах), у яких вони бідують, буквально з першого дня.

– Синхронізувати кілька паролів з одним ім'ям користувача для всіх систем.

– При необхідності оперативно змінювати або відзивати права на доступ (наприклад, при переході співробітника в іншу групу або при звільненні).

– Підтримувати відповідність урядовим постановам.

У цей блок включені наступні можливості.

Самообслуговування облікового запису, що дозволяє:

– відображати структурні схеми;

– повідомляти про додатки, пов'язані з користувачем, для адміністратора;

– змінювати дані профілю;

– виконувати пошук у каталозі;

– змінювати пароль, відповідь на запит-відповідь пароля і його підказку;

– переглядати стан політики й синхронізації пароля;

– створювати облікові записи для нових користувачів і груп (при наявності відповідних повноважень).

Запити й твердження, що дозволяють:

– запитувати ресурси;

– перевіряти підтвердження запитів на ресурси;

– працювати із призначеними завданнями підтвердження інших запитів на ресурси;

– виконувати запити й твердження в якості чиєїсь довіреної особи або делегата;

– призначати кого-небудь ще довіреною особою або делегатом (при наявності відповідних повноважень);

– управляти всіма цими функціями запитів і підтверджень в інтересах Вашої групи (при наявності відповідних повноважень);

– при необхідності для кожного запиту або підтвердження надавати цифровий підпис.

Ролі, що дозволяють виконувати наступні дії:

– запитувати призначення ролей і управляти процесом підтвердження запитів на призначення ролей;

– перевіряти стан Ваших запитів ролей;

– визначати ролі і їхні взаємини;

– визначати обмеження поділу обов'язків (SoD) і управляти процесом підтвердження у випадках, коли користувач запитує перевизначення обмеження;

– переглядати довідник ролей;

– переглядати докладні звіти, у яких перераховані ролі й обмеження поділу обов'язків, визначені в довіднику, а також поточний стан призначення ролей, виключення поділу обов'язків і повноваження користувача.

Модуль "Дотримання" дозволяє:

– Запитувати підтвердження профілю користувача.

– Запитувати підтвердження поділу обов'язків (SoD).

– Запитувати підтвердження призначення функцій.

– Запитувати підтвердження призначення користувача.

Блок менеджеру паролів. Надає можливості не тільки для простого збереження паролів, але й для повноцінної роботи з ними. Програма підтримує роботу з декількома аккаунтами, і працювати з нею можуть трохи користувачів. При цьому бази даних кожного користувача шифруються.

Додаткові можливості:

– Система пошуку по базі даних.

– Підтримка макросів.

– Можливість резервного копіювання бази даних.

– Можливість швидкого перемикання між користувачами.

– Швидкий доступ до часто використовуваних функцій.

– Генератор паролів.

– Можливість роздруківки паролів.

Блок журналювання подій. Призначений для запису у журнал усіх подій, які відбуваються у системі. Журнал дій користувачів містить форму для запуску архівації журналу. Форма архівації являє собою кнопку "Очистити журнал" і поле з датою "по:". Дату можна встановлювати будь-яку, але не раніше, ніж поточна дата мінус 1 місяць, щоб у системі завжди зберігалися дані про дії користувачів як мінімум за місяць.

Після натискання кнопки "Очистити журнал" у Системі генерується текстовий файл із архівом журналу за обраний період. Файл зберігається в зашифрованому виді, а посилання на цей файл показуються адміністраторові. Після створення файлу запису журналу за обраний період віддаляються з бази даних. У випадку помилки при створенні або збереженні файлу, записи не видаляються.

Допомога. Блок призначений про надання допомоги по роботі з системою, а також для надання інформації про розробників системи, версію та дату випуску.

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 магістерського проектування, наведена на рисунку 3.11.

Першим процесом, який запускається у системі є процес введення паролю, який взаємодіє відповідно з процесами шифрування, або дешифрування даних.

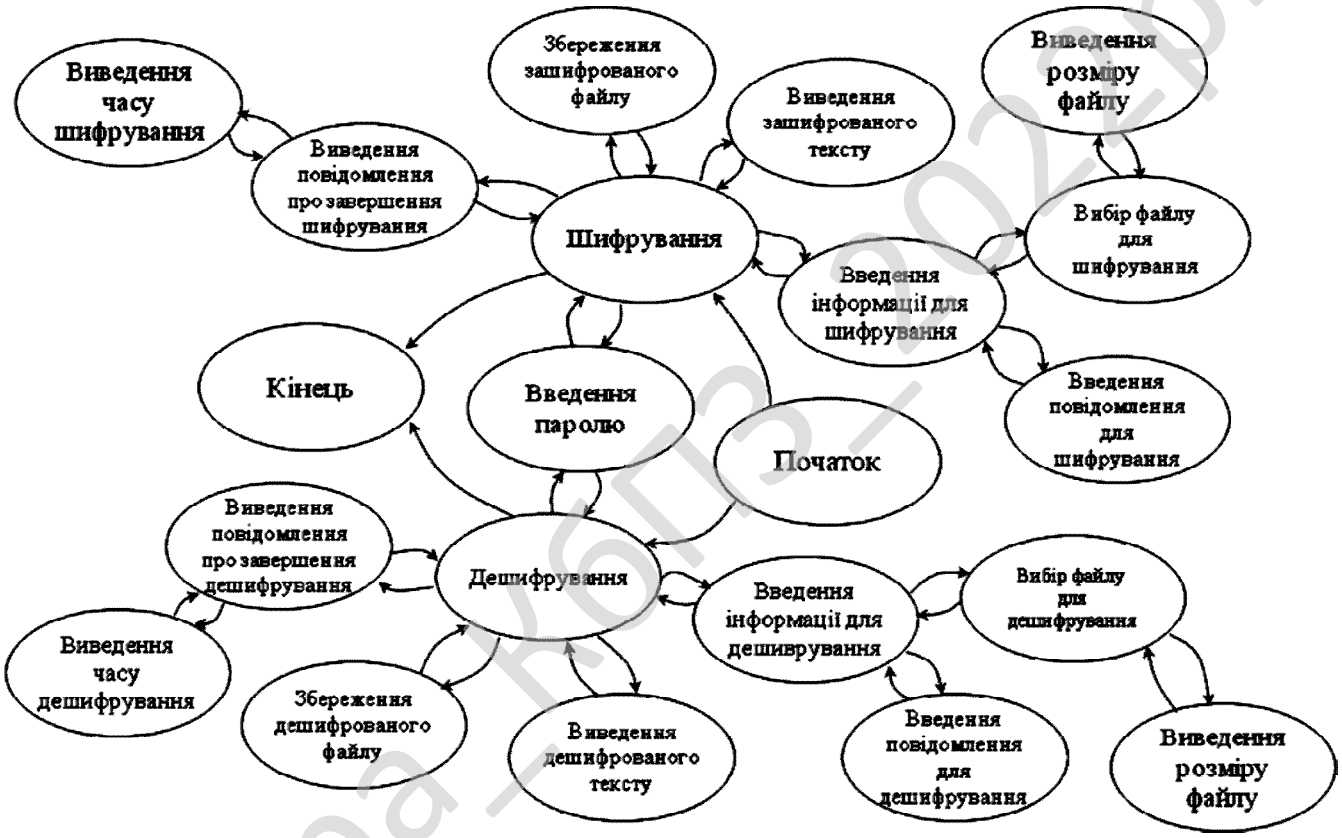

Рисунок 3.11 – Діаграма взаємодії процесів

Процес шифрування взаємодіє з наступними процесами:

– Процес виведення повідомлення про завершення шифрування, який взаємодіє з процесом виведення часу шифрування.

– Процес збереження зашифрованого файлу.

– Процес виведення зашифрованого тексту.

– Процес виведення інформації для шифрування.

Останній процес взаємодіє з наступними процесами:

– Процес введення повідомлення для шифрування.

– Процес вибору файлу для шифрування, який взаємодіє з процесом виведення розміру файлу.

Процес дешифрування взаємодіє з наступними процесами:

– Процес виведення повідомлення про завершення дешифрування, який взаємодіє з процесом виведення часу дешифрування.

– Процес збереження дешифрованого файлу.

– Процес виведення дешифрованого тексту.

– Процес виведення інформації для дешифрування.

Останній процес взаємодіє з наступними процесами:

– Процес введення повідомлення для дешифрування.

– Процес вибору файлу для дешифрування, який взаємодіє з процесом виведення розміру файлу.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього обирається дія, яку має бажання виконати користувач системи.

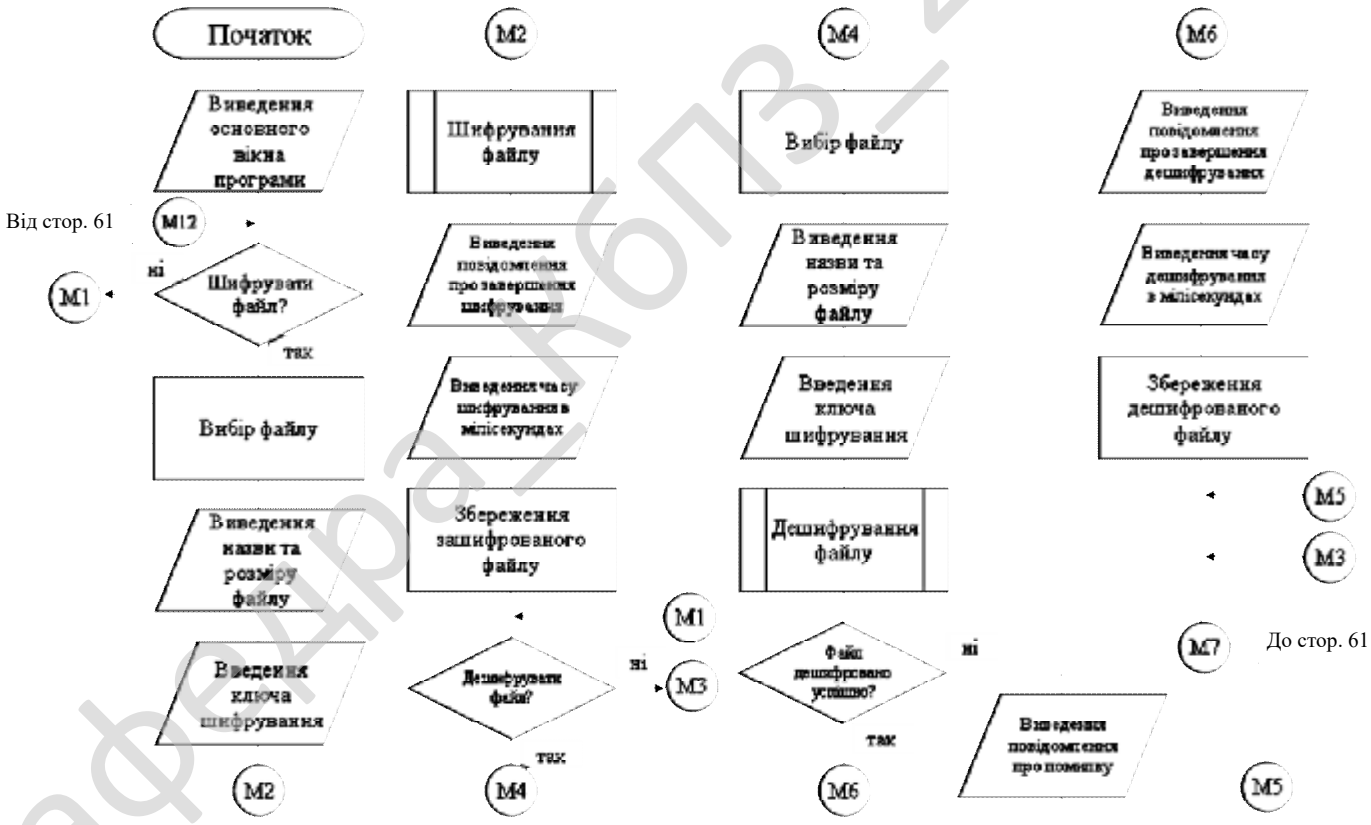

Рисунок 4.1 – Блок-схема роботи основної програми

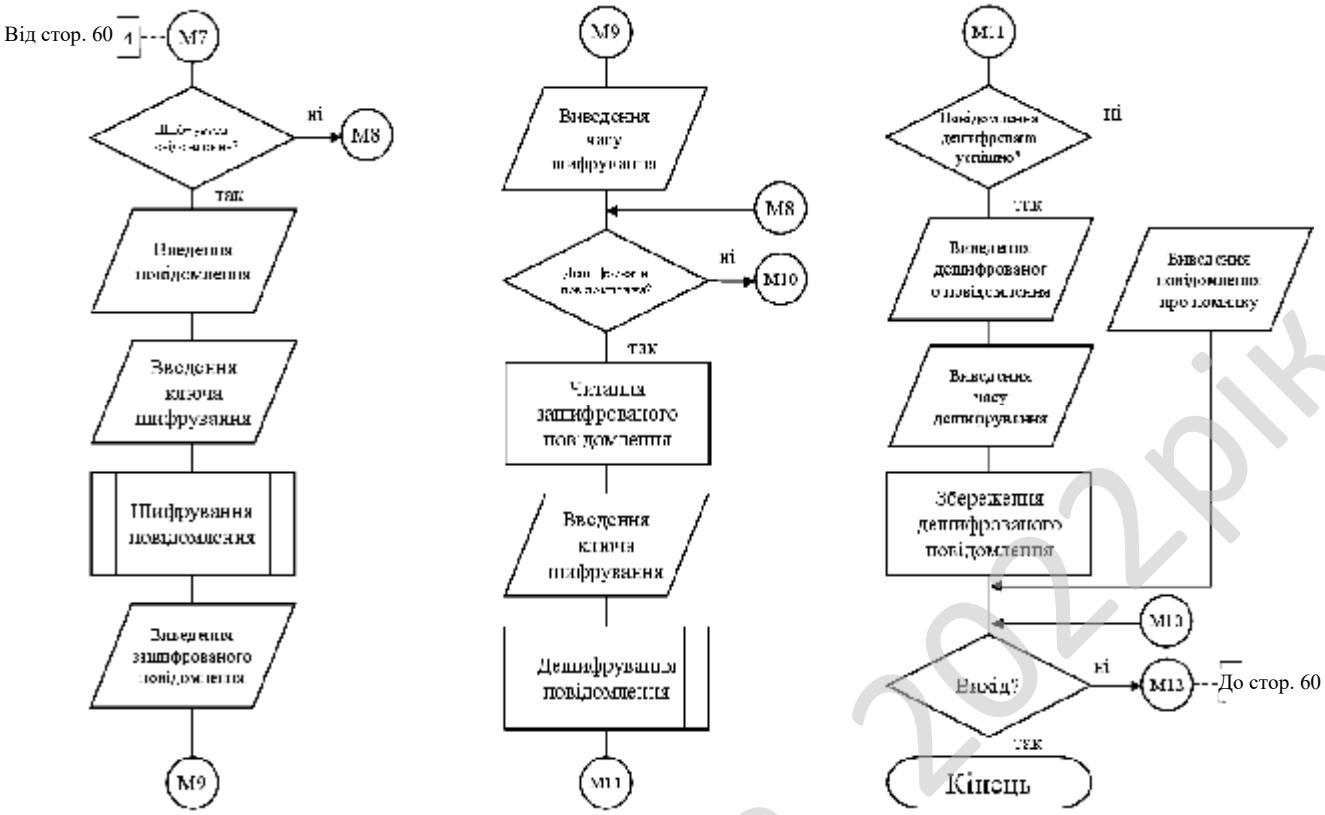

Рисунок 4.1, аркуш 2

Якщо користувач бажає шифрувати файл, то виконується наступна послідовність дій:

– Обирається файл.

– Виводиться назва та розмір файлу.

– Вводиться ключ шифрування.

– Запускається підпрограма шифрування файлу.

– Виводиться повідомлення про завершення шифрування.

– Виводиться час шифрування у мілісекундах.

– Відбувається збереження зашифрованого файлу.

Якщо користувач бажає дешифрувати файл, то виконується наступна послідовність дій:

– Обирається файл.

– Виводиться назва та розмір файлу.

– Вводиться ключ шифрування.

– Запускається підпрограма дешифрування файлу.

– Якщо файл дешифровано не успішно, то виводиться повідомлення про помилку.

– Виводиться повідомлення про завершення дешифрування.

– Виводиться час дешифрування у мілісекундах.

– Відбувається збереження дешифрованого файлу.

Якщо потрібно шифрувати повідомлення, то виконується наступна послідовність дій:

– Вводиться повідомлення.

– Вводиться ключ шифрування.

– Запускається підпрограма шифрування повідомлення.

– Виводиться повідомлення про завершення шифрування.

– Виводиться час шифрування у мілісекундах.

Якщо користувач бажає дешифрувати повідомлення, то виконується наступна послідовність дій:

– Читається зашифроване повідомлення.

– Вводиться ключ шифрування.

– Запускається підпрограма дешифрування повідомлення.

– Якщо повідомлення дешифровано не успішно, то виводиться повідомлення про помилку.

– Виводиться повідомлення про завершення дешифрування.

– Виводиться час дешифрування у мілісекундах.

– Відбувається збереження дешифрованого повідомлення.

На цьому програма закінчує свою роботу.

На рисунку 4.2 зображено блок-схему роботи підпрограми шифрування/дешифрування алгоритмом AES

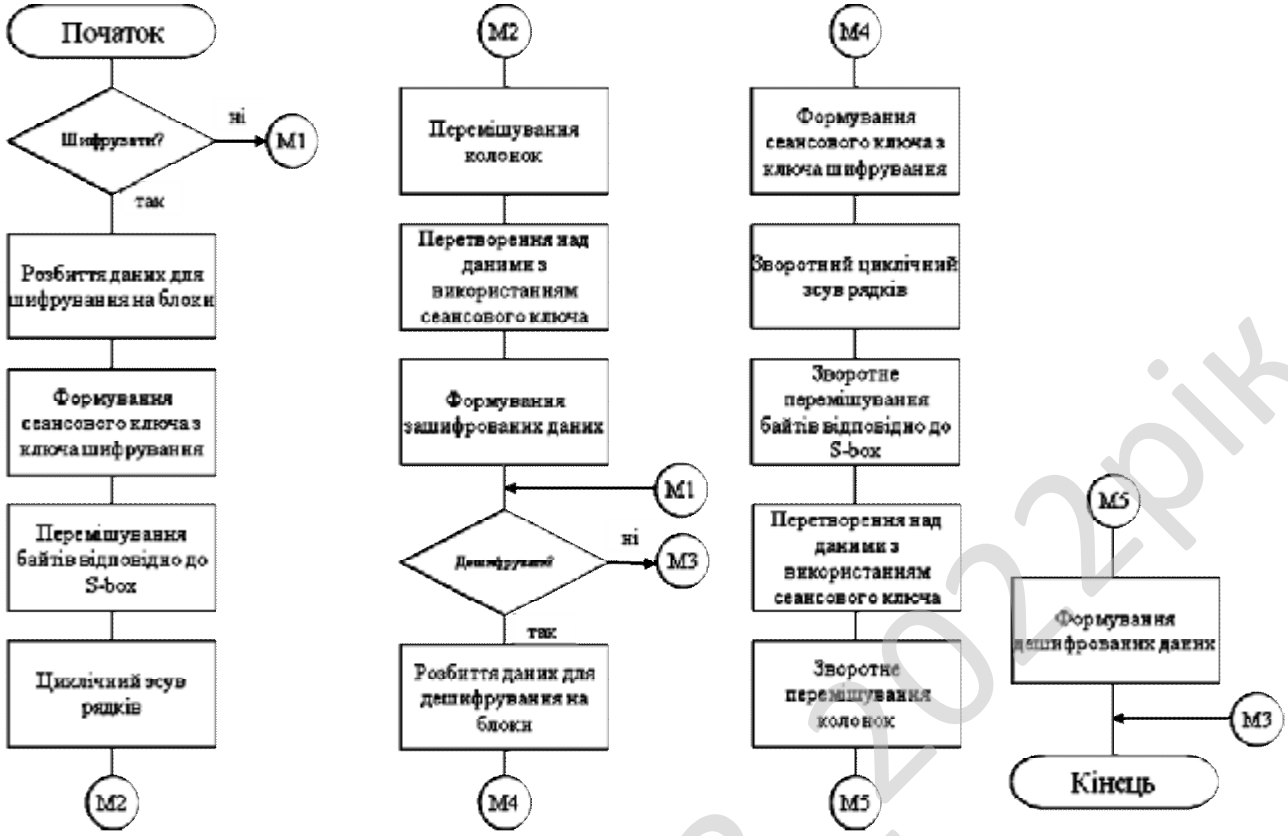

Рисунок 4.2 – Блок-схема роботи підпрограми шифрування/ дешифрування алгоритмом AES

Якщо потрібно шифрувати дані, тоді необхідно виконати наступні кроки:

– Розбити дані для шифрування на блоки.

– Формується сеансовий ключ з ключа шифрування.

– Перемішуються байти відповідно до S-box.

– Рядки циклічно зсуються.

– Колонки перемішуються.

– Відбуваються перетворення даних з використанням сеансового ключа.

– Формуються зашифровані дані.

Якщо потрібно дешифрувати дані, тоді необхідно виконати наступні кроки:

– Розбити дані для дешифрування на блоки.

– Формується сеансовий ключ з ключа шифрування.

– Рядки зворотно циклічно зсуються.

– Зворотно перемішуються байти відповідно до S-box.

– Відбуваються перетворення даних з використанням сеансового ключа.

– Колонки зворотно перемішуються.

– Формуються дешифровані дані.

Приведемо частину коду розробленої програми, яка шифрує та дешифрує дані.

```
//Процедура шифрування
procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer);
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;
//-----------------------------------------------------------------------------
//Процедура дешифрування
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer);
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;
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути

легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'а даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish

стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо використовує xor підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими

словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі система. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Верхнього меню; Розділу обрання групи шифрування файлів; Розділу введення та виведення результату роботи системи у текстовому вгляді.

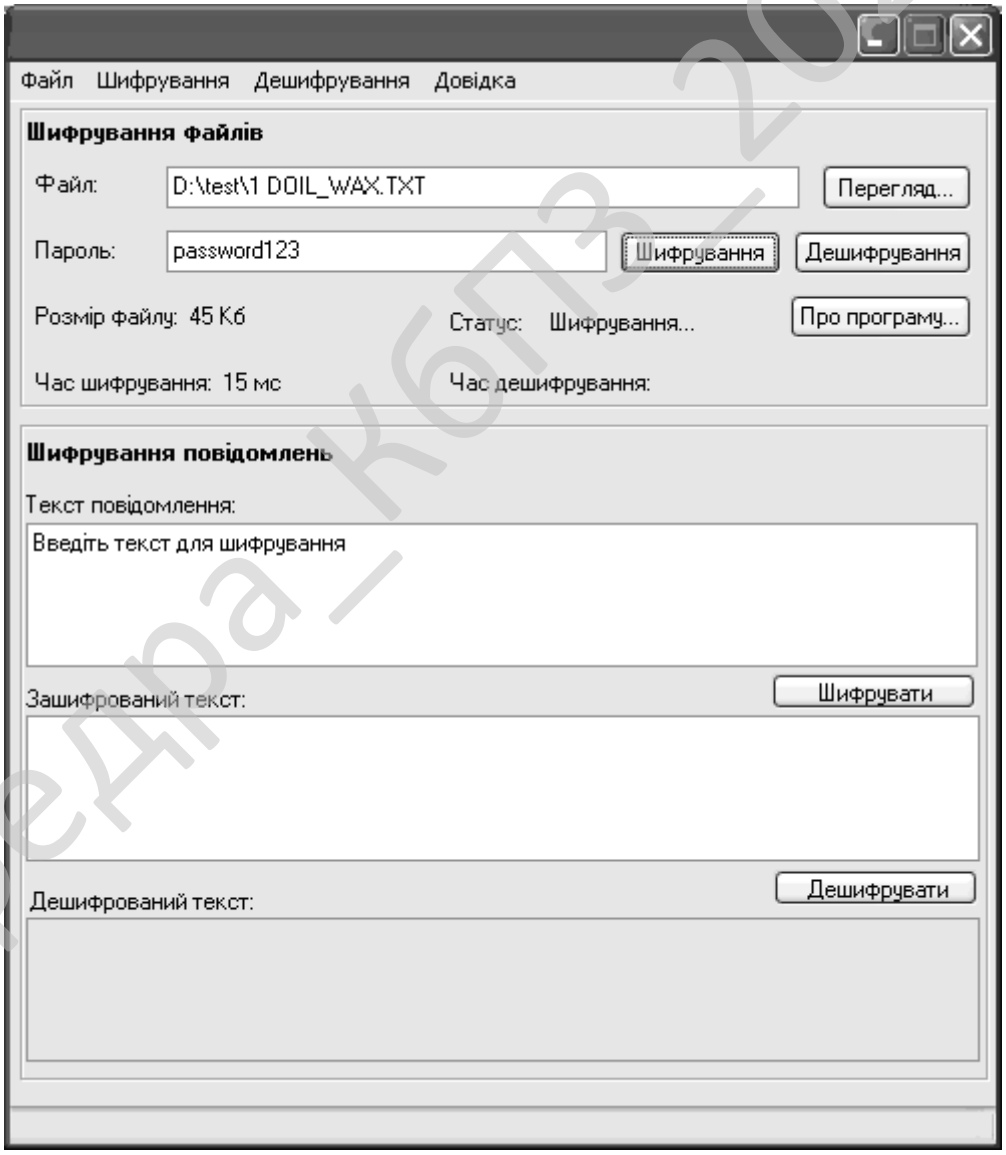

Рисунок 5.1 – Шифрування файлу

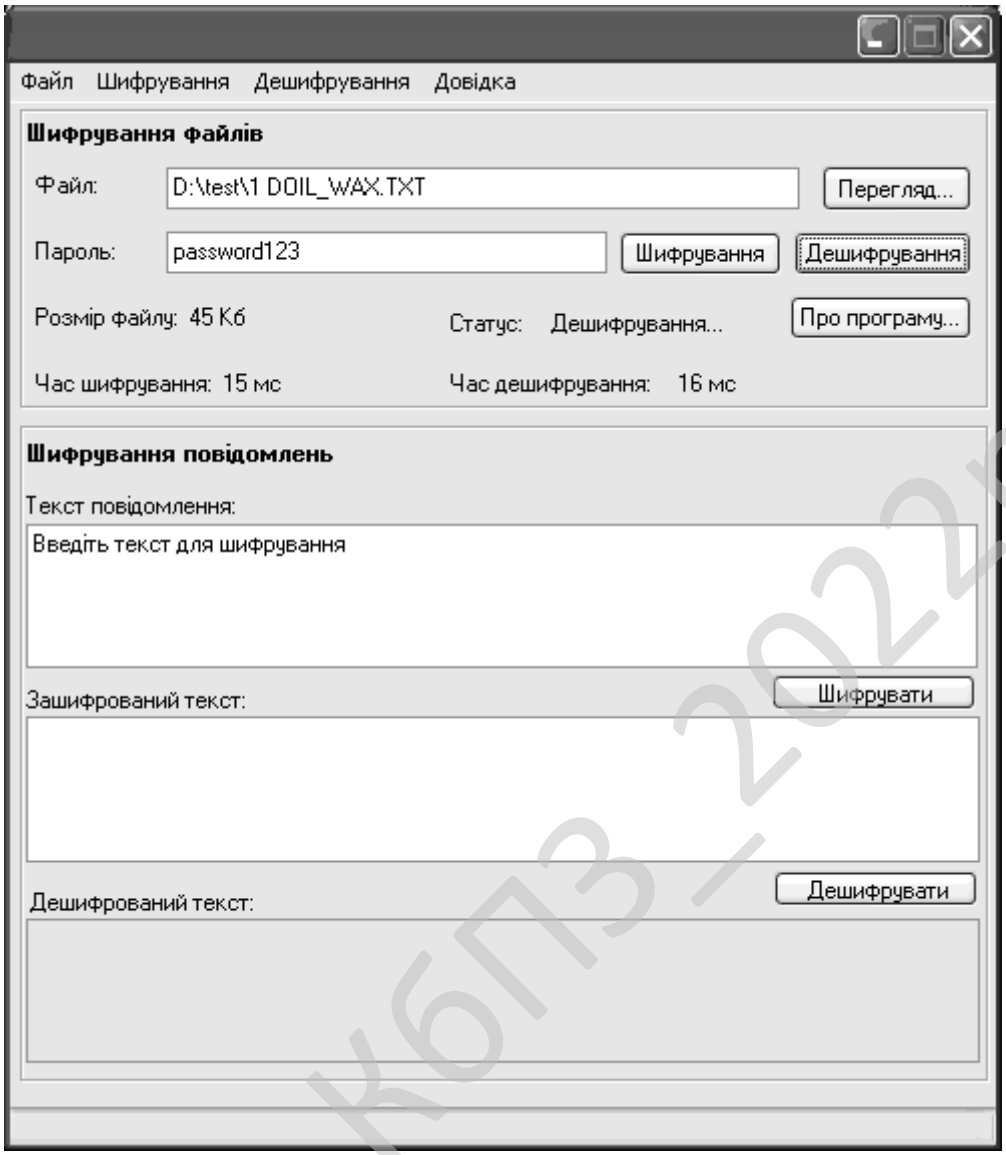

Рисунок 5.2 – Дешифрування файлу

На рисунку 5.2 зображено інтерфейс вікна дешифрування. На рисунку 5.3 зображен приклад шифрування повідомлень. На рисунку 5.4 зображено приклад дешифрування повідомлень. На рисунку 5.5 зображено довідку, у якій вказано ким розроблявся магістерський проект, хто керівник, і місце розробки.

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows 10 без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення

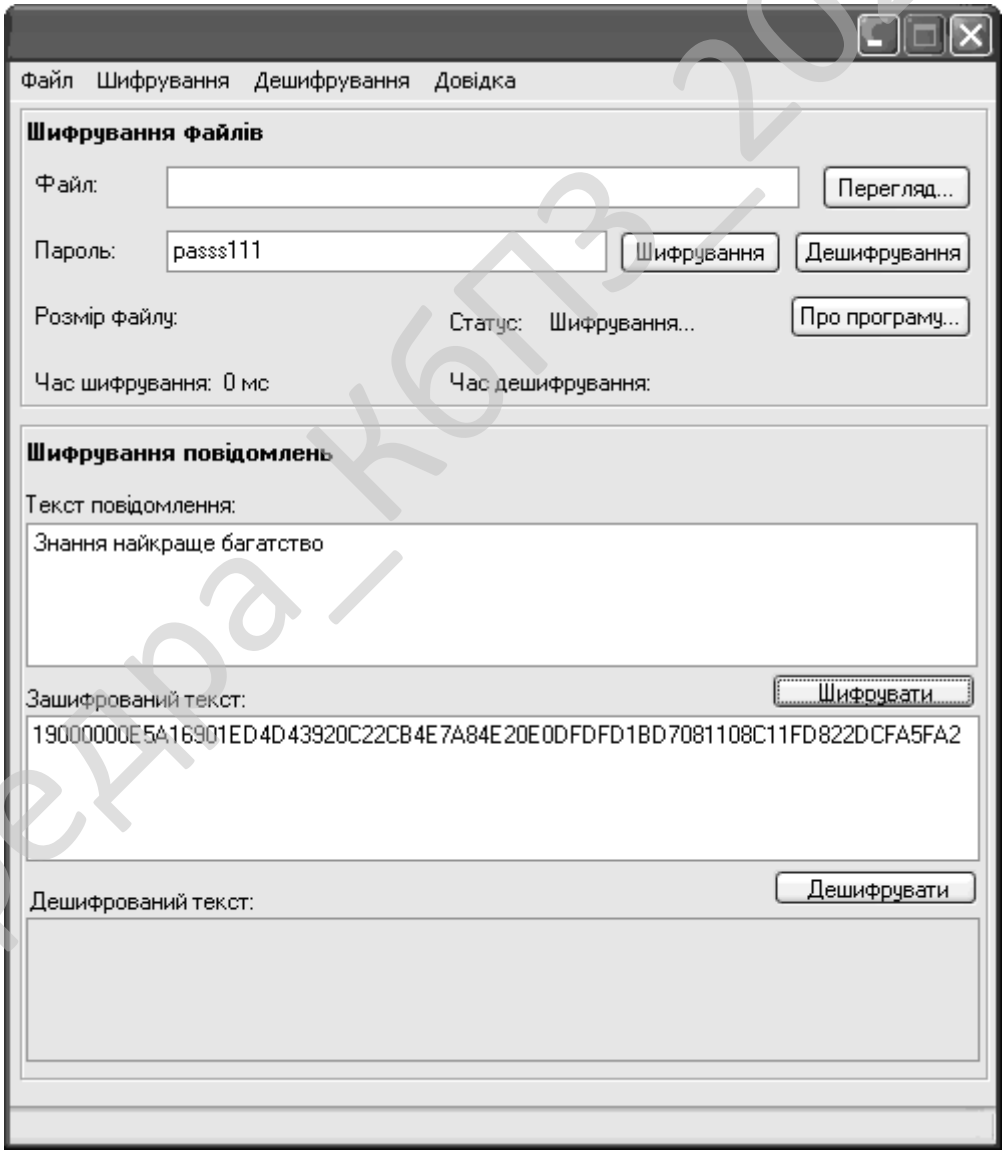

Рисунок 5.3 – Шифрування повідомлень

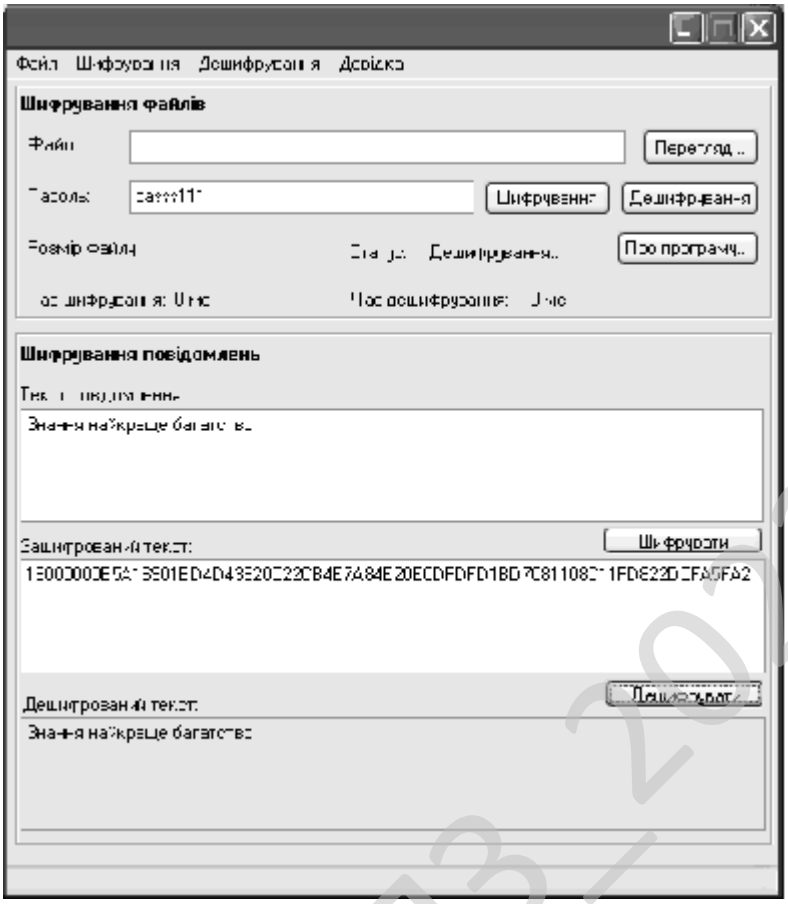

Рисунок 5.4 – Дешифрування повідомлень

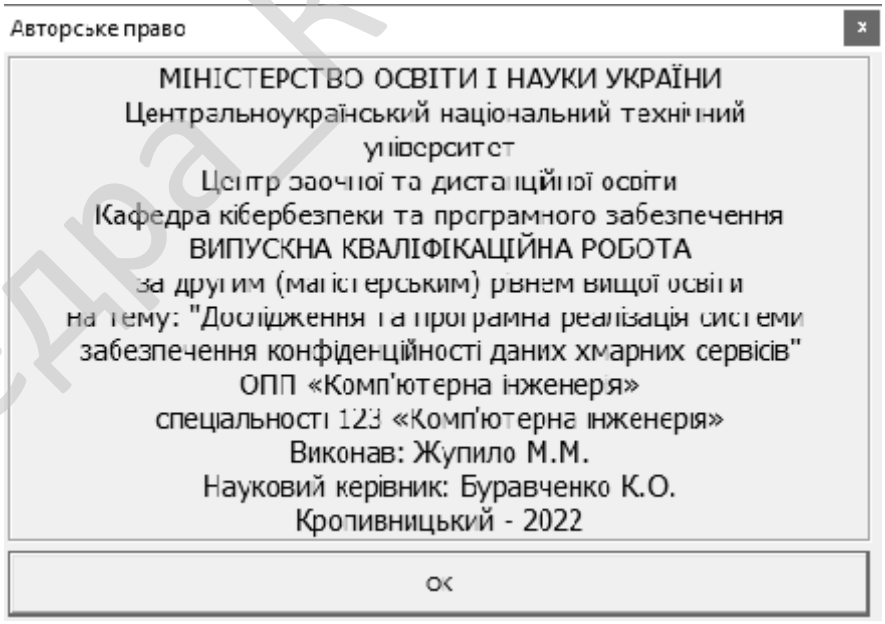

Рисунок 5.5 – Довідка

6 НАУКОВА НОВИЗНА

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

Метою розробки є дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

Об'єктом дослідження є процес забезпечення конфіденційності даних хмарних сервісів.

Предметом дослідження є методи забезпечення конфіденційності даних хмарних сервісів.

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

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

– Удосконалено метод забезпечення конфіденційності даних хмарних сервісів.

– Розроблено вітчизняний продукт забезпечення конфіденційності даних хмарних сервісів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	280
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	28000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

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

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

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

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

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

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

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

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

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

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 44 = 90 \text{ люд/день.}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	90	Ф 7.1-7.4
Впровадження	13	Д13
Всього	131	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{131 \cdot 1}{60 - 5} = 2,4 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	9	810	13,5
Монітор	60	9	540	9
Клавіатура	30	9	270	4,5
Маніпулятор «мишка»	30	9	270	4,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	48,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{49 \cdot 3}{1,2} = 122,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 122,5/(60 \cdot 8) = 0,2 \text{ ставки.}$$

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

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

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

Продовження таблиці 7.4

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

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,25	14000	10500
Продакт-менеджер	0,25	13128	9846
Інженер-програміст	2,4	13000	93600
Інженер-електронщик	0,2	12000	7200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	14000	21000
Всього за період розробки	$R_{сп}$ = 5,1	-	$Ф_{роб}$ = 196146

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{196146}{5{,}1 \cdot 60} = 641 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

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

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Компбест за 06.10.22 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i5-4590 (4 ядра по 3.3-3.7GHz), MB cache	1750
Системна плата	MSI H81M-P33 6 x USB 2.0, 2 x USB 3.(VGA, DVI, 2 x PS/2, LAN (RJ-45), 5 x Audic FireWire	1200
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	750
Жорсткий диск	240 GB SSD	1200
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC(Registered (M393B1K70CH0-CH9Q5)	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

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

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

$$З_о = 641 \cdot 131/280 = 300 \text{ грн.}$$

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

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

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

$$З_д = 300 \cdot 10 \cdot 0{,}01 = 30 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(300+30) = 73 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

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

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

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

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

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

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

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д \, , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 33,6 грн./шт.

$$З_{М2} = 100 \cdot 33{,}6 = 3360 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з ., \quad (7.18)$$

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

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

$$З_М = (105 + 3360 + 1702)/280 = 18 \text{ грн.}$$

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

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

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

$$О_п = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 280 прим.):

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

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

$$A_м = 190286 \cdot 3/(280 \cdot 12) = 170 \text{ грн.}$$

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

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

$$C_п = 300+30+73+45+18+45+170 = 681 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	300
2. Додаткова зарплата виконавців	$З_д$	30
3. Відрахування на соціальні потреби	$C_{оц}$	73
4. Загальногосподарські витрати	$Г_{осп}$	45
5. Витрати на матеріали	$З_М$	18
6. Освоєння нових операційних систем, мов програмування	$О_п$	45
7. Амортизація основних фондів	$A_м$	170
8. Повна собівартість програмного забезпечення	$C_п$	681
9. Плановий прибуток	$П_р$	341
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	1022
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	204,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1226,4

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 681 = 341 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1226
Всього капітальних витрат	–	1226

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	27500	7500
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	28988	8837

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення витрати на обслуговування системи зменшились з 27500 грн до 7500 грн на рік.

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	1226	–	306,5
Всього відрахувань	-	–	1226	–	306,5

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

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

$$З_{ел\ баз} = 0{,}545 \cdot 1300 \cdot 2{,}1 = 1487{,}85 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 900 \cdot 2{,}1 = 1030{,}05 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

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

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

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

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

$$Ев = (1022 - 681) \cdot 280 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 28000) \cdot 3/12 = 50208 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}6 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (28988\text{-}8837)\text{-}0{,}25 \cdot 1226 = 19845 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	280
2. Повна собівартість розробленої програми	Грн.	681
3. Ціна розробленої програми	Грн.	1022
4. Плановий прибуток від реалізації розробленої програми	Грн.	341
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	95480
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	50208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,6
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	19845
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

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

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

$$T_{сп} = \frac{1226}{28988 - 8837} = 0{,}1 \; року \, .$$

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – система збереження життя і здоров'я працівників у процесі трудової діяльності, що включає правові, соціально-економічні, організаційні, технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

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

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

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

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

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

з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2].

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

Ці шкідливі фактори можуть привести до професійних захворювань.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової діяльності. Їх праця стала більш інтенсивною, напруженою і вимагає значних витрат розумової, емоційної і фізичної енергії. Це призвело до необхідності у знаходженні комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку. Охорона здоров'я робітників, забезпечення безпеки умов праці, ліквідація та профілактика професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

8.2 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі

будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов'язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об'єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об'єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має

бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об'єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

8.3 Характеристика умов праці програміста

В приміщенні, в якому проводиться розробка і дослідження програмного продукту, відсутні умови, які можуть створювати підвищену або особливо підвищену небезпеку, тому воно відноситься до класу звичайних приміщень згідно Правил улаштування електроустановок (ПУЕ). Джерелом живлення є трифазна мережа напруги 380/220 В з глухо заземленою нейтралі, з частотою 50 Гц згідно За пожежо-вибухонебезпекою приміщення відноситься до класу В. В таблиці 8.2 наведена загальна характеристика приміщення щодо вибухопожеженобезпеки та важкістю робіт.

Температура повітря в приміщенні визначається температурою зовнішнього повітря і тепловою енергією, що виділяється всередині приміщення. Джерелами теплоти в даному приміщенні є люди, електроустаткування, а також освітлювальні прилади в темний час доби. Зовнішнім джерелом надлишкового тепла є сонячна радіація у світлий час доби. Робота, виконувана в даному приміщенні, відноситься до категорії I-а. Людиною в цьому випадку виділяється до 120 ккал теплової енергії в годину. Вологість повітря в приміщенні визначається вологістю атмосферного і видихуваного людьми повітря, а також випарами з поверхні шкіри.

Таблиця 8.1 – Загальна характеристика приміщення щодо вибухопожежонебезпеки та важкістю робіт

Характеристика приміщень за вибухопожежною категорією та класом зони	Загальна характеристика приміщення	Категорія за важкістю робіт згідно ГН 3.3.5-8.6.6.1 -2002
В – пожежонебезпечне клас П – II	Звичайне без ознак хімічного забруднення та нормальної вологості і за санітарними нормами	1а…….до 139 Вт/м^2 1б…….до 140-174 Вт/м^2 Клас умов праці – оптимальний

У приміщенні немає виділення шкідливих газів. Тому що в ньому не проводиться монтажних робіт, пайки чи інших робіт, при яких виділяються шкідливі гази.

Для нормалізації параметрів повітряного середовища також періодично здійснюється провітрювання приміщення і вологе прибирання. У всьому будинку діє встановлена загально обмінна витяжна вентиляція.

Раціональне освітлення приміщення сприяє кращому виконанню виробничого завдання і забезпеченню комфорту при роботі. Для забезпечення нормального освітлення застосовуються природне, однобічне, бічне і штучне освітлення, а також сполучене, нормуються зідно ДБН В.2.5-28-2006 Природне і штучне освітлення [1].

За результатами виміру освітленості величина освітленості від систем загального штучного освітлення дорівнює 310 лк, що відповідає вимогам, як пред'являються до даного приміщення.

Основними джерелами шуму на робочих місцях , обладнаних відео дисплейними терміналами, є принтер, сканер факс і обладнання для кондиціювання повітря, в самих відео дисплейних терміналів – вентилятори систем охолодження і трансформатори.

Згідно ДСанПіН 3.3.2.007-98 [2] допустимий еквівалентний рівень шуму для робочого місця програміста складає 50 дБА (акустичних децибела).

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [4, 10, 11].

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються ветикальні електроди: метелевий куток *63·63·6 мм*., (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною *L=2 м.,* та горизонтальний електрод – металева полоса з перетином *60·5 мм*. Напруга – *220/380 В*.

Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: $H=0,5$ *м*. Відстань між вертикальними заземлювачами (електродами) $A=2$ *м*. Глибина закладення горизонтального контура заземлення $t=0,6$ *м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,6+2/2=1,6 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідной климатичной зони у багатошаровому грунті [12];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [12].

Еквівалентний діаметр ветикальньного електрода (кутка) [12]:

$$Dв=0,95 \cdot K = 0,95 \cdot 63 = 59,85 \text{ мм.} = 0,0598 \text{ м.}$$

де $K = 63$ *мм.* – розмір метелевого кутка (задан).

Відношення $A/L=2/2=1$

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [12]:

$$R_O=0,366(\rho/L)[lg(2L/Dв)+(1/2)lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/2)[(lg(2 \cdot 2/0,0598)+(1/2)lg((4 \cdot 1,6+2)/(4 \cdot 1,6-2))]=19,6 \text{ Ом.}$$

Визначаємо коефіцієнт екранування ветикальньних електродів *Кев=0,62* при орієтовній кількості ветикальньних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом*:

$$N=R_О /(Кев\ R_{ЗН})= 19,6 / (0,62\cdot4)= 7,8 \approx 8\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot8=16,6\approx 16\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [12]:

$$R_П =0,366(\rho\cdot K_П/L_П)lg(2\cdot L_П^2 /(B\cdot t))=$$

$$=0,366(40\cdot5/16)\cdot lg((2\cdot16^2)/(0,06\cdot0,6))=20,2\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичної зони для з'єднуючої полоси [12]:

$B = 60\ мм. = 0,06\ м.$- ширина з'єднуючої полоси (задана).

Загальньний опір розтіканню електричного струму заземлювача [12]:

$$R =(R_О\cdot R_П)/(R_О\cdot\eta_П+ N\cdot R_П\cdot Кев)=$$

$$=(19,6\cdot 20,2)/(19,6\cdot 0,6+ 8\cdot 20,2\cdot 0,62)=3,53\ Ом.$$

де $\eta_П = 0,6$ – табличне значення коефіцієнта екранування з'єднуючої полоси [12].

Умова $R \le R_{ЗН}$ виконується ($3,53 \le 4$).

Остаточно отримали: кількість вертикальних електродів дорівнює *8* при:

$$R = 3,53\ Ом.$$

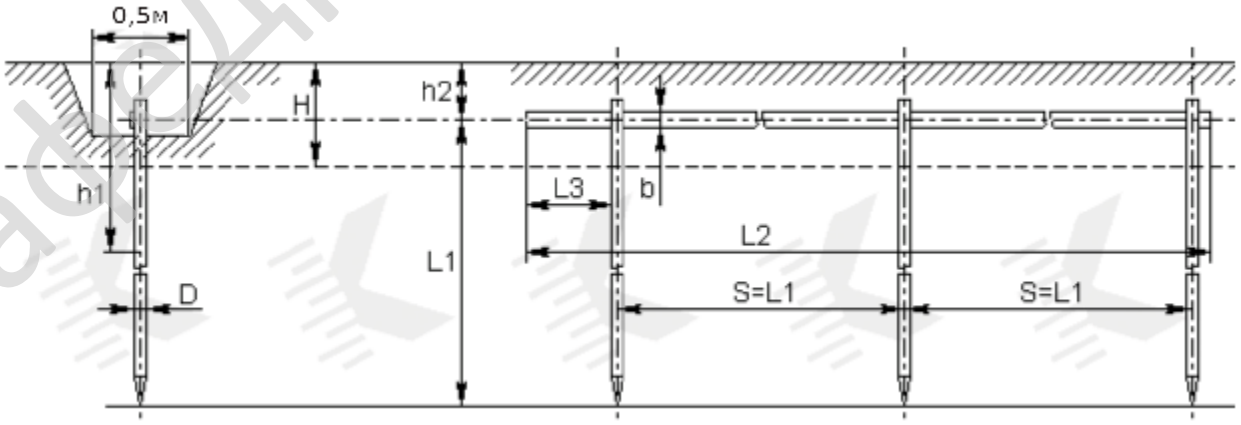

Рисунок 8.1 – Схема штучного заземлення

8.6 Висновки до розділу

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

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

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

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів забезпечення конфіденційності даних хмарних сервісів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем забезпечення конфіденційності даних хмарних сервісів.

– Досліджена система забезпечення конфіденційності даних хмарних сервісів.

– На основі отриманих результатів досліджень створена програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання забезпечення конфіденційності даних хмарних сервісів.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 19845 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Жупило М.М. Дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Menezes A., van Oorschot P., Vanstone S. Handbook of Applied Cryptography// http://www.cacr.math.uwaterloo.ca – CRC Press, 1996.

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

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

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

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

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

8. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk

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

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

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

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

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

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

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

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

16. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions».

Lecture Notes in Networks and Systems, vol 152. Springer, Cham. 2021, pp 66-84. (Scopus).

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

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

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

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

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

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

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

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

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

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

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

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

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

29. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

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

31. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання

технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х.: ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

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

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

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

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

36. 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.

37. 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.

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

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

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

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

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

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

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

45. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко,

А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

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

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

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

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

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

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

52. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з

використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

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

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

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

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

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

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

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

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

61. Постанова № 42 від 01.12.1999 Головного державного санітарного

лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0086.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Жупило М.М.			Дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Буравченко К.О.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М3	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи забезпечення конфіденційності даних хмарних сервісів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 20-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи забезпечення конфіденційності даних хмарних сервісів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи забезпечення конфіденційності даних хмарних сервісів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 125аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 23.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Буравченко К.О.

Дослідження та програмна реалізація

системи забезпечення конфіденційності даних хмарних сервісів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 51

Літера: РП

Кропивницький – 2022 року

Файл Main.pas – основна програма

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ElAES, 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: TAESKey128;
  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;
      EncryptAESStreamECB(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;
      DecryptAESStreamECB(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: TAESKey128;
```

```
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;
    EncryptAESStreamECB( 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: TAESKey128;
  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...
    DecryptAESStreamECB(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 ElAES.pas – шифрування/дешифрування алгоритмом AES

```
unit ElAES;

interface

uses
  Classes, SysUtils;

type
  EAESError = class(Exception);

  PInteger  = ^Integer;

  TAESBuffer = array [0..15] of byte;
  TAESKey128 = array [0..15] of byte;
  TAESKey192 = array [0..23] of byte;
  TAESKey256 = array [0..31] of byte;
  TAESExpandedKey128 = array [0..43] of longword;
  TAESExpandedKey192 = array [0..53] of longword;
  TAESExpandedKey256 = array [0..63] of longword;

  PAESBuffer =^TAESBuffer;
  PAESKey128 =^TAESKey128;
  PAESKey192 =^TAESKey192;
  PAESKey256 =^TAESKey256;
  PAESExpandedKey128 =^TAESExpandedKey128;
  PAESExpandedKey192 =^TAESExpandedKey192;
  PAESExpandedKey256 =^TAESExpandedKey256;

// Розширення ключа для шифрування

procedure ExpandAESKeyForEncryption(const Key: TAESKey128;
  var ExpandedKey: TAESExpandedKey128); overload;
procedure ExpandAESKeyForEncryption(const Key: TAESKey192;
  var ExpandedKey: TAESExpandedKey192); overload;
procedure ExpandAESKeyForEncryption(const Key: TAESKey256;
  var ExpandedKey: TAESExpandedKey256); overload;

// Блок раундів шифрування

procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer); overload;
procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer); overload;
procedure EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer); overload;

// Поток раундів Шифрування (ECB режим)

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream); overload;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream); overload;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream); overload;
procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream); overload;

// Поток раундів шифрування (CBC режим)

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
```

```
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

// Перетворення сеансового ключа для дешифрування

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey128);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey128;
  var ExpandedKey: TAESExpandedKey128); overload;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey192);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey192;
  var ExpandedKey: TAESExpandedKey192); overload;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey256);
overload;
procedure ExpandAESKeyForDecryption(const Key: TAESKey256;
  var ExpandedKey: TAESExpandedKey256); overload;

// Блок раундів дешифрування

procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer); overload;
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer); overload;
procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer); overload;

// Поток раундів дешифрування (ECB режим)

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream); overload;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream); overload;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream); overload;
procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream); overload;

// Поток раундів дешифрування (CBC режим)

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
```

```
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream); overload;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream); overload;
procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  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,
    $09070708E, $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 ExpandAESKeyForEncryption(const Key: TAESKey128; var ExpandedKey:
TAESExpandedKey128);
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 ExpandAESKeyForEncryption(const Key: TAESKey192; var ExpandedKey:
TAESExpandedKey192); 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 ExpandAESKeyForEncryption(const Key: TAESKey256; var ExpandedKey:
TAESExpandedKey256); 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 EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer);
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 EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer);
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 EncryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer);
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 ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey128);
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 ExpandAESKeyForDecryption(const Key: TAESKey128; var ExpandedKey:
TAESExpandedKey128);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey192);
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 ExpandAESKeyForDecryption(const Key: TAESKey192; var ExpandedKey:
TAESExpandedKey192);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

procedure ExpandAESKeyForDecryption(var ExpandedKey: TAESExpandedKey256);
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 ExpandAESKeyForDecryption(const Key: TAESKey256; var ExpandedKey:
TAESExpandedKey256);
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  ExpandAESKeyForDecryption(ExpandedKey);
end;

//Процедура дешифрування

procedure DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey128;
  var OutBuf: TAESBuffer);
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 DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey192;
  var OutBuf: TAESBuffer);
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 DecryptAES(const InBuf: TAESBuffer; const Key: TAESExpandedKey256;
  var OutBuf: TAESBuffer);
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 EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream);
var
```

```
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  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);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
```

```
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  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);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  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);
    EncryptAES(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 DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey128; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
```

```
  const ExpandedKey: TAESExpandedKey128; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey192; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const Key: TAESKey256; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
```

```
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamECB(Source, Count, ExpandedKey, Dest);
end;

procedure DecryptAESStreamECB(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256; Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError.Create(SReadError);
    DecryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Dec(Count, SizeOf(TAESBuffer));
  end;
end;

// Поток раундів шифрування (CBC режим)

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  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(TAESBuffer) 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])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
```

```
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  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])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  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(TAESBuffer) 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])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  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])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
  end;
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForEncryption(Key, ExpandedKey);
  EncryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure EncryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut, Vector: TAESBuffer;
  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(TAESBuffer) 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])^;
    EncryptAES(TempIn, ExpandedKey, TempOut);
    Done := Dest.Write(TempOut, SizeOf(TempOut));
    if Done < SizeOf(TempOut) then
      raise EStreamError.Create(SWriteError);
    Vector := TempOut;
    Dec(Count, SizeOf(TAESBuffer));
  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])^;
    EncryptAES(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 DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey128; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey128;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;

procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey128;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(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(TAESBuffer));
  end;
end;


procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey192; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey192;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;


procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey192;  const InitVector: TAESBuffer;
```

```
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do


begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(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(TAESBuffer));
  end;
end;


procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const Key: TAESKey256; const InitVector: TAESBuffer; Dest: TStream);
var
  ExpandedKey: TAESExpandedKey256;
begin
  ExpandAESKeyForDecryption(Key, ExpandedKey);
  DecryptAESStreamCBC(Source, Count, ExpandedKey, InitVector, Dest);
end;


procedure DecryptAESStreamCBC(Source: TStream; Count: cardinal;
  const ExpandedKey: TAESExpandedKey256;  const InitVector: TAESBuffer;
  Dest: TStream);
var
  TempIn, TempOut: TAESBuffer;
  Vector1, Vector2: TAESBuffer;
  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(TAESBuffer)) > 0 then
    raise EAESError.Create(SInvalidInBufSize);
```

```
  Vector1 := InitVector;
  while Count >= SizeOf(TAESBuffer) do
  begin
    Done := Source.Read(TempIn, SizeOf(TempIn));
    if Done < SizeOf(TempIn) then
      raise EStreamError(SReadError);
    Vector2 := TempIn;
    DecryptAES(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(TAESBuffer));
  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(' МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add(' на  тему: ');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи забезпечення
конфіденційності даних хмарних сервісів ');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Керівник: Буравченко К.О.');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Розробив: студент Жупило Микола Миколайович');
Memo1.Lines.Add('                  гр. КІ-21МЗ');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Кропивницький 2022');
Memo1.Lines.Add('');
end;

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