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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI”

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

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

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

Тарковському Данилу Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI*

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
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Тарковський Д.І.
(прізвище та ініціали)

АНОТАЦІЯ

Тарковський Д.І. Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Метою розробки є дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Об'єктом дослідження є процес визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Предметом дослідження є методи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Методи дослідження базуються на методах захисту інформації та штучного інтелекту (AI), методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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

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

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

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: комп'ютерні науки, стійкість, конфіденційність, штучний інтеллект

ABSTRACT

Tarkovskyi D.I. Research and software implementation of a system for determining the level of stability of privacy services based on AI methods. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the system of determining the level of stability of privacy services based on AI methods.

The purpose of the development is research and software implementation of a system for determining the level of stability of privacy services based on AI methods.

The object of the study is the process of determining the level of stability of privacy services based on AI methods.

The subject of the study is methods of determining the level of stability of privacy services based on AI methods.

Research methods are based on methods of information protection and artificial intelligence (AI), methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the system for determining the level of stability of privacy services based on AI methods.

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

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

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

The program was developed in the Delphi 10.4.1 environment.

Keywords: computer science, sustainability, privacy, artificial intelligence

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 10
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 13
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 13
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 31
2.3 Розгорнута постановка завдання 37
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 38
3.1 Опис функціонування системи 38
3.2 Розробка структурної схеми..... 68
3.3 Розробка функціональної схеми 69
3.4 Розробка діаграми процесів..... 72
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 74
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 74
4.2 Захист розробленого програмного забезпечення..... 82
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 83
6 НАУКОВА НОВИЗНА 88

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ..... 89
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 89
7.2 Розрахунок трудомісткості розробки програмної продукції..... 91
7.3 Визначення чисельності виконавців і планового фонду зарплати..... 93
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника..... 98
7.5 Визначення собівартості розробки та ціни програмної продукції..... 102
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 105
7.7 Визначення експлуатаційних витрат..... 105
7.8 Визначення економічної ефективності програмної продукції..... 107
7.9 Висновок..... 109
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ..... 110
8.1 Вступ..... 110
8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером..... 111
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста..... 113
8.4 Розробка заходів з умов поліпшення охорони праці..... 115
8.5 Розрахункова частина..... 116
8.6 Висновки до розділу..... 118
9 ОСНОВНІ ВИСНОВКИ..... 119
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 121

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДВЧ	–	датчик випадкових чисел
ДСТ 28147-89	–	алгоритм шифрування
ЕОМ	–	електронна обчислювальна машина
ІС	–	інформаційна система
ОС	–	обчислювальна система
ПВЧ	–	псевдовипадкові числа
ПЗ	–	програмне забезпечення
РПЗ	–	руйнуючі програмні засоби
СЗІ	–	система захисту інформації
ASCII	–	система кодування
DES	–	алгоритм шифрування
FEAL	–	алгоритм шифрування
IDEA	–	алгоритм шифрування
KOI-8	–	система кодування
RISC	–	архітектура процесора

ВСТУП

Актуальність теми. Магістерська робота присвячена проектуванню програмного забезпечення, призначеного для аналізу стійкості криптографічних систем, різних шифрів. Криптографія досліджує область захисту інформації. Основними напрямками криптографічних алгоритмів є передача конфіденційної інформації по каналам зв'язку (наприклад електронна пошта), установлення дійсності переданих повідомлень. Криптографічні системи застосуються практично в будь-яких областях пов'язаних з обробкою інформації.

Криптоаналіз призначений для того, щоб визначити надійність того або іншого шифру, перевірити його стійкість. Цілями криптоаналізу насамперед є знаходження уразливих місць у розповсюджених системах захисту даних, а також визначення надійності й трудомісткості розкриття закритої інформації, та розкриття зашифрованої інформації без повного набору необхідних даних.

Криптоаналіз і криптографія мають протилежні напрямки і є одними з підмножин криптології.

Звичайний, лінійний або диференціальний криптоаналіз або складний у реалізації, або вузько орієнтований, або не підходить по іншим міркуванням (час взлому, витрати на взлом й т.д.). З іншої сторони через складність використовуваних при аналізі співвідношень доводиться вдаватися до допомоги різних діаграм і таблиць, які часто не охоплюють весь спектр використовуваних значень, і не можуть бути перелічені.

Тому для рішення завдання зручно використовувати структуру, що самонавчається, яка залежно від вихідних даних і ступеня навченості, буде вирішувати, яким образом діяти в тій або іншій ситуації. При цьому параметри окремих елементів нейрокомп’ютерної мережі (персептронів), визначаються вагами на входах і виходах. Ще одним достоїнством цієї системи є те, що для аналізу не потрібно використовувати різні алгоритми. Безсумнівним достоїнством даного підходу є також те, що всі сучасні системи захисту свідомо уразливі до подібним до методів криптоаналізу. Також великим достоїнством

системи такого роду є висока швидкість розкриття. На аналіз однієї точки характеристики звичайними методами в досвідченого фахівця може піти до одного робочого дня. Методика для аналізу шифрів може бути реалізована у вигляді програмного забезпечення, хоча створення подібного програмного забезпечення сильно утруднено різного виду факторами (зацікавленість розроблювачів, замовників, трудомісткість реалізації й т.д.). Новизна такого підходу відіграє більшу роль. Більшість криптосистем, що вважаються досить надійними, були розроблені ще в 70-х – 80-х роках. Була розроблена безліч систем, безсумнівно більш надійних, але утримуючих у собі застарілу основу (DES, багаторазовий DES, DESX) і тому уразливих для нових методів.

Більшість відомих експертів-криптоаналітиків вважають що, штучний інтелект, тим більше нейрокомп'ютерні мережі, не кращий інструмент криптоаналізу. Підстав у цієї широко поширеної думки дуже багато, наприклад, низька можливість для навчання.

Недоліками названого підходу є, насамперед те, що для кожної нової криптографічної системи необхідно розробляти нову методику навчання елементів нейромережі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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

– Огляд існуючих систем визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– Дослідження системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– Програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Об'єктом дослідження є процес визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Предметом дослідження є методи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Методи дослідження базуються на методах захисту інформації та штучного інтелекту (AI), методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– Розроблено вітчизняний продукт визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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 визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI, 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

Останні десятиліття характеризується різким збільшенням числа відкритих робіт із всіх питань криптології, а криптоаналіз стає однією з областей досліджень, що найбільше активно розвиваються. Багато криптосистем, стійкість яких не викликала особливих сумнівів, виявилися успішно розкритими. При цьому розроблений великий арсенал математичних методів, що представляють прямий інтерес для криптоаналітика.

Перш ніж визначати задачи криптоаналізу, визначемо, яким вимогам повинні відповідати криптосистеми.

Вимоги до криптографічних систем

Процес криптографічного закриття даних може здійснюватися як програмно, так і апаратно. Апаратна реалізація відрізняється істотно більшою вартістю, однак їй властиві й переваги: висока продуктивність, простота, захищеність і т.д. Програмна реалізація більш практична, допускає відому гнучкість у використанні.

Для сучасних криптографічних систем захисту інформації сформульовані наступні загальноприйняті вимоги:

– зашифроване повідомлення повинне піддаватися читанню тільки при наявності ключа;

– число операцій, необхідних для визначення використаного ключа шифрування по фрагменту шифрованого повідомлення й відповідного йому відкритого тексту, повинне бути не менше загального числа можливих ключів;

– число операцій, необхідних для розшифровування інформації шляхом перебору всіляких ключів повинне мати строгу нижню оцінку й виходити за межі можливостей сучасних комп'ютерів (з урахуванням можливості використання

мережних обчислень), або вимагати неприйнятно високих витрат на ці обчислення;

– знання алгоритму шифрування не повинне впливати на надійність захисту;

– незначна зміна ключа повинна приводити до істотної зміни виду зашифрованого повідомлення навіть при шифруванні того самого вихідного тексту;

– незначна зміна вихідного тексту повинне приводити до істотної зміни виду зашифрованого повідомлення навіть при використанні того самого ключа;

– структурні елементи алгоритму шифрування повинні бути незмінними;

– додаткові біти, що вводяться в повідомлення в процесі шифрування, повинні бути повністю й надійно сховані в шифрованому тексті;

– довжина шифрованого тексту не повинна перевершувати довжину вихідного тексту;

– не повинне бути простих і легко встановлюваних залежностей між ключами, послідовно використовуваними в процесі шифрування;

– будь-який ключ із безлічі можливих повинен забезпечувати надійний захист інформації;

– алгоритм повинен допускати як програмну, так і апаратну реалізацію, при цьому зміна довжини ключа не повинна вести до якісного погіршення алгоритму шифрування.

На початку 1970-х рр. була відома тільки класична одноключова криптографія, але число відкритих робіт із цієї тематики було досить скромним. Відсутність інтересу до неї можна пояснити цілим рядом причин. По-перше, гострої потреби в криптосистемах комерційного призначення, очевидно, ще не відчувалося. По-друге, великий обсяг закритих досліджень по криптографії утрудняв роботу багатьох учених, яким, природно, хотілося одержати нові результати. І, нарешті, може бути, найважливіший фактор полягає в тому, що

криптоаналіз як наукова дисципліна фактично як і раніше являв собою великий набір розрізнених "трюків", не об'єднаних стрункою математичною концепцією.

В 1970-х рр. ситуація радикально змінилася. По-перше, з розвитком мереж зв'язку й повсюдним вторгненням ЕОМ необхідність у криптографічному захисті даних стала усвідомлюватися усе більше широкими шарами суспільства. По-друге, винахід Діффи й Хелманном криптографії з відкритим ключем створило сприятливий ґрунт для задоволення комерційних потреб у таємності, усунувши такий істотний недолік класичної криптографії, як складність поширення ключів. По суті, цей винахід гальванізував наукове співтовариство, відкривши якісно нову недосліджену область, що до того ж обіцяв можливість широкого впровадження нових результатів швидко, що розвивається теорії, обчислювальної складності для розробки конкретних систем із простим математичним описом. Очікувалося, що стійкість таких систем буде надійно опиратися на нерозв'язність у реальному часі багатьох добре відомих завдань і що, може бути, згодом удасться довести принципове нерозкривання деяких криптосистем. Але надії на досягнення доказової стійкості за допомогою відношення завдань криптографії до добре відомих математичних завдань не виправдалася, а, скоріше, навпаки. Саме та обставина, що будь-яке завдання відшукання способу розкриття деякої конкретної криптосистеми можна переформулювати як привабливе математичне завдання, при рішенні якої вдається використовувати багато методів тої ж теорії складності, теорії чисел і алгебри, привело до розкриття багатьох криптосистем.

На сьогоднішній день класична стрічка однократного використання залишається єдиною, безумовно, стійкою системою шифрування. Ідеальний доказ стійкості деякої криптосистеми з відкритим ключем могло б складатися в доказі того факту, що будь-який алгоритм розкриття цієї системи, що володіє малою ймовірністю її розкриття, якої неможливо зневажати, пов'язаний з неприйнятно великим обсягом обчислень. І хоча жодна з відомих систем з відкритим ключем не задовольняє цьому сильному критерію стійкості, ситуацію не слід розглядати як абсолютно безнадійну. Було розроблено багато систем, у відношенні яких

доведено, що їхня стійкість еквівалентна складності рішення деяких важливих завдань, які майже всіма розглядаються як украй складні, таких, наприклад, як відома задача розкладання цілих чисел.. Крім того, результати широких досліджень, що проводилися протягом останніх десяти років як у самій криптографії, так і в загальній теорії обчислювальної складності, дозволяють сучасному криптоаналітику набагато глибше зрозуміти, що ж робить його системи нестійкими.

Проведення криптоаналізу для давно існуючих і криптоалгоритмів, що недавно з'явилися дуже актуально, тому що вчасно можна сказати, що даний криптоалгоритм нестійкий, і вдосконалити його або замінити новим. Для того щоб виявляти нестійкі криптоалгоритми необхідно увесь час удосконалювати вже відомі методи криптоаналізу й знаходити нові.

Таким чином розробка систем аналізу стійкості існуючих систем є актуальною задачею, яка потребує вирішення у даній магістерській роботі.

1.2 Область застосування

Визначимо область застосування системи, яка розробляється. Сфера інтересів *криптоаналізу* – дослідження можливості розшифровування інформації без знання ключів. Основні напрямки використання криптографічних методів – передача конфіденційної інформації з каналів зв'язку (наприклад, електронна пошта), установлення дійсності переданих повідомлень, зберігання інформації (документів, баз даних) на носіях у зашифрованому виді.

Отже, криптографія дає можливість перетворити інформацію таким чином, що її прочитання (відновлення) можливо тільки при знанні ключа.

Введемо ряд понять, які необхідні для роботи над задачею визначеною в меті дослідження. Як інформація, що підлягає шифруванню й розшифруванню, будуть розглядатися *тексти*, побудовані на деякому *алфавіті*. Під цими термінами розуміється наступне.

Алфавіт – кінцева безліч використовуваних для кодування інформації знаків.

Текст – упорядкований набір з елементів алфавіту.

Як приклади алфавітів, використовуваних у сучасних ІС можна привести:

– алфавіт $\boldsymbol{Z}_{256}$ – символи, що входять у стандартні коди ASCII і КОІ-8;

– двійковий алфавіт – $\boldsymbol{Z}_2 = \{0,1\}$;

– восьмеричний або шістнадцятковий алфавіт.

Шифрування – процес перетворення вихідного тексту, що носить також назву *відкритого тексту*, у *шифрований текст*.

Розшифрування – процес, зворотний шифруванню. На основі ключа шифрований текст перетвориться у вихідний.

Криптографічна система являє собою сімейство $\boldsymbol{T}$ перетворень відкритого тексту. Члени цього сімейства індексуються, або позначаються символом k; параметр k звичайно називається *ключем*. Перетворення T_k визначається відповідним алгоритмом і значенням ключа k.

Ключ – інформація, необхідна для безперешкодного шифрування й розшифрування текстів.

Простір ключів $\boldsymbol{K}$ – це набір можливих значень ключа. Звичайно ключ являє собою послідовний ряд букв алфавіту.

Криптосистеми підрозділяються на *симетричні* й *асиметричні* (або *з відкритим ключем*).

У *симетричних криптосистемах* для шифрування, і для розшифрування використовується той самий ключ.

У *системах з відкритим ключем* використовуються два ключі – *відкритий* і *закритий* (*секретний*), які математично зв'язані один з одним. Інформація шифрується за допомогою відкритого ключа, що доступний всім бажаючим, а розшифровується за допомогою закритого ключа, відомого тільки одержувачеві повідомлення.

Терміни *розподіл ключів* і *керування ключами* відносяться до процесів системи обробки інформації, змістом яких є вироблення й розподіл ключів між користувачами.

Електронним цифровим підписом називається приєднуване до тексту його криптографічне перетворення, що дозволяє при одержанні тексту іншим користувачем перевірити авторство й дійсність повідомлення.

Криптостійкістью називається характеристика шифру, що визначає його стійкість до розшифрування без знання ключа (тобто криптоаналізу). Є кілька показників криптостійкості, серед яких:

– кількість всіх можливих ключів;

– середній час, необхідне для успішної криптоаналітичної атаки того або іншого виду.

Ефективність шифрування з метою захисту інформації залежить від збереження таємниці ключа й криптостійкості шифру.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI, 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

Поняття стійкості криптографічного алгоритму

Прийнято розрізняти криптоалгоритми по ступеню довідності їхньої безпеки. Існують наступні види крипто алгоритмів:

– безумовно стійкі;

– доказово стійкі;

– приблизно стійкі криптоалгоритми.

Безпека безумовно стійких криптоалгоритмів заснована на доведених теоремах про неможливість розкриття ключа.

Стійкість доказово стійких криптоалгоритмів визначається складністю рішення добре відомого математичного завдання, що намагалися вирішити багато математиків і яка є загальновизнано складною.

Приблизно стійкі криптоалгоритми засновані на складності рішення приватного математичного завдання, що не зводиться до добре відомих завдань і яку намагалися вирішити один або кілька людей.

На жаль, безумовно стійкі криптосистеми незручні на практиці (системи з разовим використанням ключа вимагають великої захищеної пам'яті для зберігання ключів, системи квантової криптографії вимагають волоконо-оптичних каналів зв'язку і є дорогими, крім того, доказ їхньої безпеки йде з області математики в область фізики).

Достоїнством доказово стійких алгоритмів є гарна вивченість завдань, покладених у їхню основу. Недоліком їх є неможливість оперативної доробки криптоалгоритмів у випадку появи такої необхідності, тобто твердість цих

криптоалгоритмів. Підвищення стійкості може бути досягнуто збільшенням розміру математичного завдання або його заміною, що, як правило, тягне ланцюг змін не тільки в шифрованій, але й суміжній апаратурі.

Приблизно стійкі криптоалгоритми характеризуються порівняно малою вивченістю математичного завдання, але зате мають велику гнучкість, що дозволяє не відмовлятися від алгоритмів, у яких виявлені слабкі місця, а проводити їхню доробку.

Завдання забезпечення захищеного зв'язку містить у собі цілий комплекс проблем. Це завдання забезпечення таємності й імітозахисту, упізнавання (автентифікації) і завдання керування ключами, включаючи їхнє вироблення, розподіл і доставку користувачам, а також їхню оперативну заміну якщо буде потреба.

Джерело повідомлень виробляє довільну інформацію (відкриті тексти) з якимось розподілом ймовірностей. Шифратор шифрує це повідомлення на конфіденційному (відомому тільки відправникові й одержувачеві) ключі Z і перекладає відкритий текст у шифрований текст або шифрограму (криптограму, шифротекст). Ключі виробляються джерелом ключів і по безпечних каналах розсилаються абонентом мережі зв'язку. Дешифратор розкриває прийняту шифрограму й передає одержувачеві. Рандомізатор робить всі шифрограми несхожими одну на одну, навіть якщо вхідні повідомлення однакові. Вирішальний пристрій ухвалює рішення щодо тому, чи є прийняте повідомлення справжнім, тобто виконує функцію імітозахисту. Операції шифрування й розшифрування можна описати так: $Y = E(X)$, $X = D(Y)$.

Для взаємної однозначності необхідно, щоб DE було одиничним перетворенням. Передбачатися наявність у відправника й одержувача загального секретного ключа Z. (Насправді, ключі в них не обов'язково однакові, але знання одного ключа, наприклад шифрування, дозволяє легко обчислити іншої. Тому розглянуті криптоалгоритми іноді називають симетричними, або одноключовими. Відповідно, і криптографія, що займається вивченням таких

алгоритмів, називається одноключовою). Дана схема застосовується в тому випадку, якщо абоненти мережі довіряють один одному.

Дійсність і цілісність повідомлення забезпечуються його криптографічним перетворенням, виконуваним за допомогою секретного ключа. Наприклад, якщо відправник передасть відразу й відкритий (не потребуючий засекречування), і зашифрований тексти, то це дозволить одержувачеві, що знає ключ, стверджувати, що текст при передачі по каналу зв'язку не був змінений, якщо результат розшифрування шифрограми збігається з відкритим текстом. Дійсно, випадковий збіг відповідних один одному відкритого тексту й шифрограми – практично неможлива подія. Цю пару міг скласти лише відправник, що знає секретний ключ. Звідси треба й дійсність повідомлення (відправник ототожнюється із власником ключа). У дійсності немає необхідності передавати всю шифрограму, досить передати лише її частину, називану імітовставкою, що повинна залежати від усього відкритого тексту. Тоді одержувач може на підставі отриманого тексту й ключа обчислити свою імітовставку й перевірити її відповідність отриманій.

Для впізнавання користувача використовується наступний діалог. Той, хто перевіряє виробляє випадковий текст і посилає тому, хто опізнає, для шифрування. Той. хто опізнає, шифрує цей текст і повертає тому, хто що перевіряє. Той перевіряє відповідність шифрограми тексту. Правильну відповідь може скласти тільки власник секретного ключа, що ототожнюється із законним користувачем. Очевидно, що порушник не зможе правильно зашифрувати новий текст і назватися користувачем. (Виключенням є аналогія відомого шахрайства, застосованого при грі в шахи поштою, коли порушник просто транслює відповіді й запити справжнім перевіряючим й перевіряємому, ведучи діалог одночасно з кожним з них). Принципова відмінність даної системи від упізнавання по паролю, де підслуховування дозволяє довідатися секретний пароль і надалі скористатися цим, полягає в тому, що тут по каналі зв'язку секретна інформація не передається. Ясно, що й стійкість шифрування, і імітостійкість, і стійкість

упізнавання можуть бути забезпечені, якщо покладене в основу криптоперетворення є стійким у змісті розкриття ключа.

Криптографічні алгоритми звичайно будуються з використанням простих і швидко виконуваних операторів декількох типів. Безліч оборотних операторів, що перетворять текст довжиною n біт у текст довжиною n біт, є елементами групи оборотних операторів по множенню (підстановок n-розрядних слів).

Нехай f,g,h – оборотні оператори, тобто існують f^{-1}, g^{-1} , h^{-1}. Тому hgf – послідовне виконання операторів f,g,h – теж оборотний оператор (оператори виконуються праворуч ліворуч) зі зворотним оператором до цього добутку f^{-1}, g^{1}, h^{-1} . Тому дешифратор виконує ті ж операції, що й шифратор, але у зворотному порядку, і кожний оператор розшифрування є зворотним до відповідного оператора шифрування. Деякі оператори є взаємно зворотними, тобто виконання підряд два рази деякої операції над текстом дає вихідний текст. У термінах теорії груп це записується рівнянням $f^{2} = e$, де e – одиничний оператор. Такий оператор називається інволюцією. Можна сказати, що інволюція являє собою корінь із одиниці. Прикладом інволюції є додавання по модулю два тексту із ключем.

Порушник може вирішувати наступні завдання. Він може намагатися розкрити зашифровану інформацію, організувати вибіркове пропущення тої або іншої інформації, нарешті, він може намагатися змінити справжню або нав'язати помилкову інформацію. Принципове розходження завдань засекречування й імітозахисту полягає в тому, що ключ засекречування повинен бути не доступний порушникові протягом строку таємності інформації, що звичайно набагато більше, ніж термін дії ключа й може становити десятки років. Ключ імітозахисту становить інтерес для порушника тільки під час його дії. Тому й вимоги до нього пред'являються менш тверді, ніж до ключа засекречування.

Існує ще одне важливе застосування одноключової криптографії. Це здійснення розраховуємого в одну сторону перетворення інформації. Таке перетворення називається геш-функцією. Особливість цього перетворення

полягає в тому, що пряме перетворення $y=h(x)$ обчислюється легко, а зворотне $x= h^{-1}(y)$ – важко. Загалом кажучи, зворотне перетворення не є функцією, тому вірніше говорити про знаходження одного із прообразів для даного значення геш-функції. У цьому випадку ключа, що розуміється як деяка конфіденційна інформація, немає. Однак стійкі геш-функції, для яких прообраз за даним значенням функції важко знайти, реалізуються криптографічними методами й вимагають для обґрунтування стійкості проведення криптографічних досліджень. Типове застосування геш-функції – створення стислого образа для вихідного тексту такого, що знайти інший текст, що володіє в такий же спосіб, розрахунково неможливо. Завдання створення стійкої геш-функції виникають, наприклад, при цифровому підписі текстів.

Одне з можливих самостійних застосувань геш-функцій – це впізнавання користувача за допомогою ланцюжка виду x , $h(x)$, $h(h(x)) = h^2(x), h^3(x), \ldots h^k(x)$.

Останнє значення ланцюжка $h^k(x)$ є контрольною інформацією для того, хто перевіряє, а користувач знає $h^{k-1}(x)$ і пред'являє цю інформацію з вимоги того, хто перевіряє. Той, хто перевіряє обчислює $h(h^{k-1}(x))$ і порівнює з контрольної. Наступного разу цей користувач повинен пред'явити $h^{k-2}(x)$, а контрольною інформацією є $h^{k-1}(x)$ і т.д. Це цікаве рішення, запропоноване А. Конхеймом, однак має ряд недоліків. По-перше, користувачеві треба зберігати весь ланцюжок $h^i(x)$, що вимагає великого обсягу пам'яті, якщо число впізнавань може бути велике. По-друге, якщо в кожного користувача є декілька перевіряючих, то встає питання про синхронізацію останнього використаного значення, тих. хто *перевіряють по показниках,* h^i(x), тобто потрібні канали зв'язку між кожною парою що перевіряють.

Здатність криптосистеми протистояти атакам (активного або пасивного) криптоаналітика називається стійкістю. Кількісно стійкість виміряється як складність найкращого алгоритму, що приводить криптоаналітика до успіху із прийнятною ймовірністю. Залежно від цілей і можливостей криптоаналітика міняється й стійкість. Розрізняють стійкість ключа (складність розкриття ключа

найкращим відомим алгоритмом), стійкість безключового читання, імітостійкість (складність нав'язування помилкової інформації найкращим відомим алгоритмом) і ймовірність нав'язування помилкової інформації. Це іноді зовсім різні поняття, не зв'язані між собою. Деякі криптосистеми, дозволяють нав'язувати помилкову інформацію зі складністю, що практично не залежить від стійкості ключа. Аналогічно можна розрізняти стійкість власно криптоалгоритму, стійкість протоколу, стійкість алгоритму генерації й поширення ключів.

Рівень стійкості залежить від можливостей криптоаналітика й від користувача. Так, розрізняють криптоаналіз на основі тільки шифрованого тексту, коли криптоаналітик розташовує тільки набором шифрограм і не знає відкритих текстів, і криптоаналіз на основі відкритого тексту, коли криптоаналітик знає й відкриття, і відповідні шифровані тексти. Оскільки криптоалгоритм звичайно повинен бути досить універсальним, природним представляється вимога, щоб стійкість ключа не залежала від розподілу ймовірностей джерела повідомлень. У загальному випадку джерело повідомлень може виробляти “зручні” для порушника повідомлення, які можуть стати йому відомими. У цьому випадку говорять про криптоаналіз на основі спеціально обраних відкритих текстів. Очевидно, що стійкість ключа стосовно аналізу на основі обраних текстів не може перевищувати стійкості стосовно аналізу на основі відкритих текстів, а вона, у свою чергу, не може перевищувати стійкості стосовно аналізу на основі шифрованих текстів. Іноді розроблювачем СЗІ допускається навіть, що ворожий криптоаналітик може мати доступ до криптосистеми, тобто бути «своїм». Звичайно криптоалгоритми розробляють так, щоб вони були стійкими стосовно криптоаналізу на основі спеціально обраних відкритих текстів.

Поняття «найкращого алгоритму» розкриття ключа у визначенні стійкості неконструктивно й допускає суб'єктивне тлумачення (для когось із розроблювачів найкращим алгоритмом може бути простий перебір ключів). Очевидно, ні для одного з використовуваних криптоалгоритмів не визначений

найкращий алгоритм розкриття ключа, тобто завдання знаходження найкращого алгоритму є надзвичайно складною. Тому на практиці для оцінки стійкості користуються найкращим відомому або знайденим у ході досліджень алгоритмом розкриття. Таким чином, на практиці ніхто не може перешкодити здатному криптоаналітику знизити оцінку стійкості, придумавши новий, більш ефективний метод аналізу.

Створення нових ефективних методів розкриття ключа або іншого методу ослаблення криптоалгоритму може давати обізнаним особам більші можливості по завданню збитків користувачам, що застосовують даний криптоалгоритм. Публікація або замовчування цих відомостей визначаються ступенем відкритості суспільства. Рядовий користувач системи неспроможний перешкодити порушникові в розкритті його ключів.

З викладеного треба, що поняття «найкращого відомого» алгоритму неабсолютно: завтра може з'явитися новий більш ефективний алгоритм розкриття, що приведе до неприпустимого зниження стійкості криптоалгоритма. З розвитком математики й засобів обчислювальної техніки стійкість криптоалгоритма може тільки зменшуватися. Для зменшення можливого збитку, викликаного несвоєчасною заміною криптоалгоритму, що втратили свою стійкість, бажаний періодичний повторний огляд стійкості криптоалгоритма. Для зниження ймовірності непередбаченого «обвалу» знову розробленого криптоалгоритма необхідне проведення криптографічних досліджень.

З розглянутого вище видно, що поняття стійкості криптосистеми багатогранно. Стійкість залежить не тільки від розроблювача, але й від особливостей використання даного криптоалгоритма в системі керування або зв'язку, від фізичної реалізації криптоалгоритма, а також від майбутніх успіхів математики й обчислювальної техніки. Адже криптосистема може експлуатуватися багато років, а необхідність зберігати в секреті протягом тривалого часу передану раніше по відкритих каналах зв'язку інформацію може зробити необхідним прогнозувати розвиток науки й техніки на десятиліття.

Аналіз надійності криптосистем

У сучасному програмному забезпеченні (ПЗ) криптоалгоритми широко застосовуються не тільки для завдань шифрування даних, але й для автентифікації й перевірки цілісності. На сьогоднішній день існують добре відомі й апробовані криптоалгоритми (як із симетричними, так і несиметричними ключами), криптостійкість яким або доведена математично, або заснована на необхідності рішення математично складного завдання (факторизації, дискретного логарифмування й т.п.). Таким чином, вони не можуть бути розкриті інакше, чим повним перебором або рішенням зазначеного завдання.

З іншого боку, у комп'ютерному й навколокомп'ютерному світі увесь час з'являється інформація про помилки або «дирах» у тій або іншій програмі (у т.ч. що застосовує криптоалгоритми), або про те, що вона була зламана (*cracked*). Це створює недовіру як до конкретних програм, так і до можливості взагалі захистити що-небудь криптографічними методами не тільки від спецслужб, але й від простих хакерів.

Тому знання історії атак і «дир» у криптосистемах, а також розуміння причин, по яких вони мали місце, є однією з необхідних умов розробки захищених систем. Перспективним напрямком досліджень у цій області є аналіз успішно проведених атак або виявлених уразливостей у криптосистемах з метою їхнього узагальнення, класифікації й виявлення причин і закономірностей їхньої появи й існування.

За аналогією з таксономією причин порушення безпеки ОС, виділимо наступні причини ненадійності криптографічних програм:

1. Неможливість застосування стійких криптоалгоритмів;
2. Помилки в реалізації криптоалгоритмів;
3. Неправильне застосування криптоалгоритмів;
4. Людський фактор.

Причини ненадійності можна відобразити на наступній схемі:

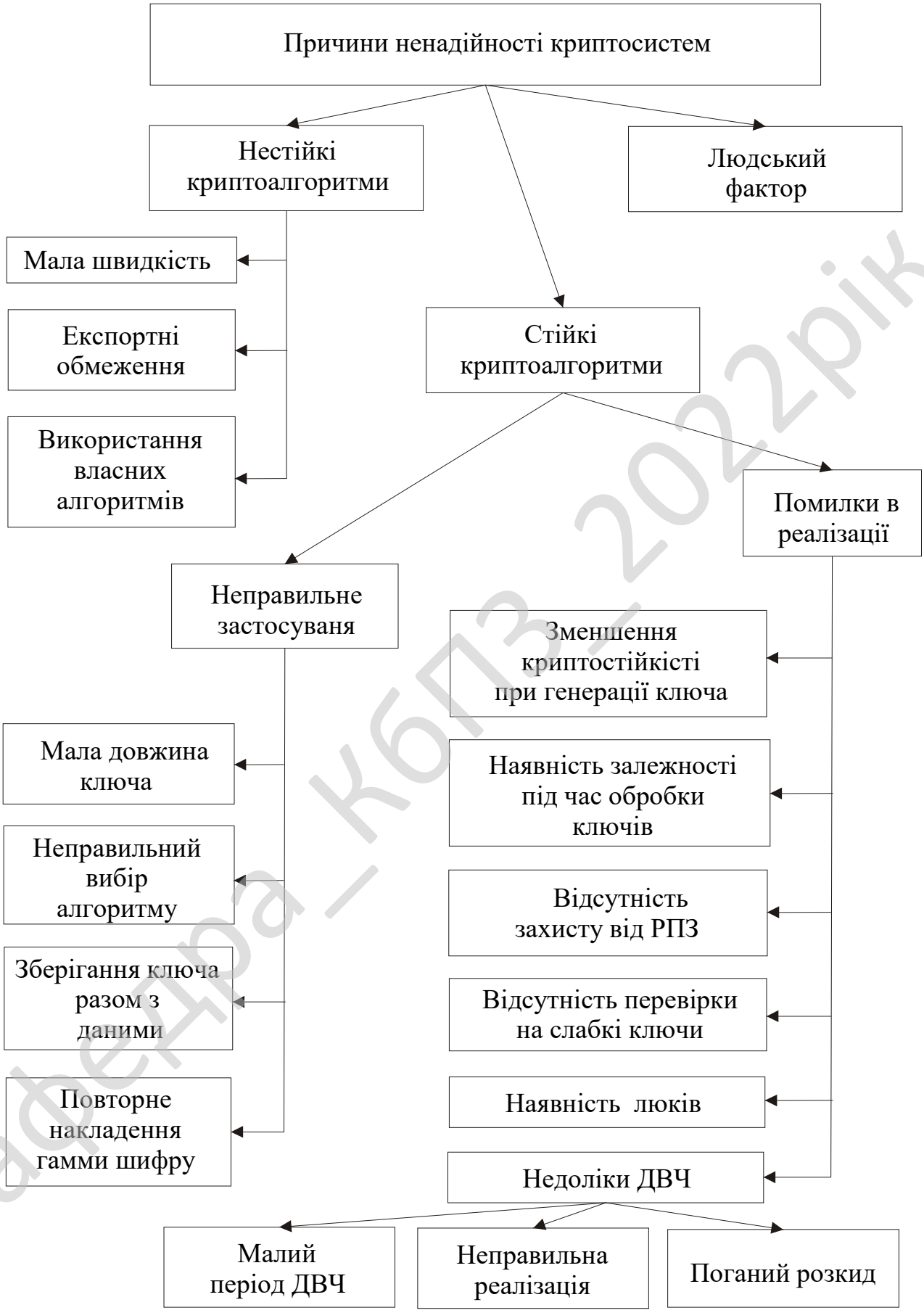

Рисунок 2.1 – Причини ненадійності криптосистем

Неможливість застосування стійких криптоалгоритмів

Ця група причин є найпоширенішою через наступних факторів.

Мала швидкість стійких криптоалгоритмів

Це основний фактор, що утрудняє застосування гарних алгоритмів в, наприклад, системах «тотального» шифрування або шифрування «на літу».

Експортні обмеження

Це причина, пов'язана з експортом криптоалгоритмів або з необхідністю здобувати патент або права на них.

Використання власних криптоалгоритмів

Незнання або небажання використовувати відомі алгоритми – така ситуація, як ні парадоксально, також має місце бути, особливо в програмах типу Freeware і Shareware, наприклад, архіваторах.

Далі, використання слабких алгоритмів часто приводить до успіху *атаки по відкритому тексті*.

Неправильна реалізація криптоалгоритмів

Незважаючи на те, що в цьому випадку застосовуються криптостійкі або сертифіковані алгоритми, ця група причин приводить до порушень безпеки криптосистем через їхню неправильну реалізацію.

Зменшення криптостійкісти при генерації ключа

Ця причина з досить численними прикладами, коли криптосистема або обрізає пароль користувача, або генерує з нього дані, що мають меншу кількість біт, ніж сам пароль.

Відсутність перевірки на слабкі ключі

Деякі криптоалгоритми (зокрема, DES, IDEA) при шифруванні з специфічними ключами не можуть забезпечити належний рівень криптостійкості. Такі ключі називають слабкими *(weak).* Для DES відомо 4 слабких і 12 напівслабких *(semi-weak)* ключів. І хоча ймовірність потрапити в них рівняється ~2L 10^{-16}, для серйозних криптографічних систем зневажати єю не можна.

Потужність безлічі слабких ключів IDEA становить не багато – не мало – 2^{51} (втім, через те, що всього ключів 2^{128}, імовірність потрапити в нього в $3L10^{7}$ разів менше, ніж в DES).

Недостатня захищеність від РПЗ

РПЗ (руйнуючі програмні засоби) – це комп'ютерні віруси, троянські коні, програмні закладки й тому подібні програми, здатні перехопити секретний ключ або самі нешифровані дані, а також просто підмінити алгоритм на некриптостійкий. У випадку якщо програміст не передбачив достатніх способів захисту від РПЗ, вони легко здатні порушити безпеку криптосистеми. Особливо це актуально для операційних систем, що не мають убудованих засобів захисту або засобів розмежування доступу – типу MS DOS або Windows 95:

1. Перехоплення пароля.
2. Підміна криптоалгоритму.
3. Троянський кінь в електронній пошті.

Наявність залежності в часі обробки ключів

Це порівняно новий аспект недостатньо коректної реалізації криптоалгоритмів. Багато криптосистем неоднаково швидко обробляють різні вхідні дані. Це відбувається як через апаратні (різна кількість тактів на операцію, влучення в процесорний кеш і т.п.), так і програмних причин (особливо при оптимізації програми за часом). Час може залежати як від ключа шифрування, так і від шифруємих даних.

Тому зловмисник, маючи детальну інформацію про реалізацію криптоалгоритма, маючи зашифровані дані, і будучи здатним якимось образом вимірювати час обробки цих даних (наприклад, аналізуючи час відправлення пакетів з даними), може спробувати підібрати секретний ключ. Причому ключ можна одержувати, уточнюючи біт за битому, а кількість необхідних вимірів часу прямо пропорційно довжині ключа.

І хоча поки не вдалося довести ці дослідження до конкретного результату (обчислити секретний ключ), цей приклад показує, що програмування систем

критичного призначення (у т.ч. і криптосистем) повинне бути особливо ретельним і, можливо, для цього необхідно застосовувати особливі захисні методи програмування й спеціалізовані засоби розробки (особливо компілятори).

Помилки в програмній реалізації

Ясно, що поки програми будуть писатися людьми, цей фактор завжди буде мати місце.

Наявність люків

Причини наявності люків у криптосистемах очевидний: розроблювач хоче мати контроль над оброблюваною в його системі інформацією й залишає для себе можливість розшифровувати її, не знаючи ключа користувача. Можливо також, що вони використовуються для налагодження й з якоїсь причини не вбираються з кінцевого продукту. Природно, що це рано або пізно стає відомим досить великому колу осіб і цінність такої криптосистеми стає майже нульовою.

Недоліки датчика випадкових чисел (ДВЧ)

Гарний, математично перевірений і коректно реалізований ДВЧ також важливий для криптосистеми, як і гарний, математично стійкий і коректний криптоалгоритм, інакше його недоліки можуть вплинути на загальну криптостійкість системи. При цьому для моделювання ДВЧ на ЕОМ звичайно застосовують датчики псевдовипадкових чисел (ПВЧ), що характеризуються періодом, розкидом, а також необхідністю його ініціалізації (*seed*). Застосування ПВЧ для криптосистем взагалі не можна визнати вдалим рішенням, тому гарні криптосистеми застосовують для цих цілей фізичний ДВЧ (спеціальну плату), або, принаймні, виробляють число для ініціалізації ПВЧ за допомогою фізичних величин (наприклад, часу натискання на клавіші користувачем).

Малий період і *поганий розкид* ставляться до математичних недоліків ДВЧ і з'являються в тому випадку, якщо з якихось причин вибирається власний ДВЧ. Інакше кажучи, вибір власного ДВЧ так само небезпечний, як і вибір власного криптоалгоритму.

У випадку малого періоду (коли псевдовипадкових значень, вироблюваних датчиком, менше, ніж можливих значень ключа) зловмисник може скоротити час пошуку ключа, перебираючи не самі ключі, а псевдовипадкові значення й генеруючи з них ключі.

При поганому розкиді датчика зловмисник також може зменшити середній час пошуку, якщо почне перебір із самих імовірних значень псевдовипадкових чисел.

Найпоширенішою помилкою, що проявляється й у випадку гарного ПВЧ, є його *неправильна ініціалізація*. У цьому випадку число, використовуване для ініціалізації, має або менше число біт інформації, чим сам датчик, або обчислюється з невипадкових чисел і може бути передвіщене з тим або іншим ступенем імовірності.

Неправильне застосування криптоалгоритмів

Ця група причин приводить до того, що виявляється ненадійними криптостійкі й коректно реалізовані алгоритми.

Мала довжина ключа

Це сама очевидна причина.

Помилковий вибір класу алгоритму

Це також досить розповсюджена причина, при якій розроблювач вибирає нехай і гарний, але зовсім невідповідний до його завдання алгоритм. Найчастіше це вибір шифрування замість гешування або вибір симетричного алгоритму замість алгоритму з відкритими ключами.

Повторне накладення гами шифру

Зберігання ключа разом з даними

Ця причина приводить до того, що дані, зашифровані за допомогою криптостійкого й коректно реалізованого алгоритму, можуть бути легко дешифровані. Це зв'язано зі специфікою розв'язуваного завдання, при якій неможливо вводити ключ ззовні й він зберігається десь усередині в практично незашифрованому вигляді. Інакше кажучи, тут найбільш уразливим буде

алгоритм шифрування не ключем, а ключа (за допомогою якогось вторинного ключа). Але так як (що знов-таки очевидно треба зі специфіки завдання) цей вторинний ключ зберігати ззовні не можна, то основні дані рано або пізно будуть розшифровані без використання методів перебору.

Людський фактор

У будь-якій критичній системі помилки людини-оператора є чи ледве не самими дорогими й розповсюдженими. У випадку криптосистем непрофесійні дії користувача зводять нанівець самий стійкий криптоалгоритм і саму коректну його реалізацію й застосування.

У першу чергу це пов'язане з вибором паролів. Очевидно, що короткі або осмислені паролі легко запам'ятовуються людиною, але вони набагато простіше для розкриття. Використання довгих і безглуздих паролів, безумовно, краще з погляду криптостійкісти, але людина звичайно не може їх запам'ятати й записує на папірці, що потім або губиться, або попадає в руки зловмисникові.

Саме з того, що недосвідчені користувачі звичайно вибирають або короткі, або осмислені паролі, існують два методи їхнього розкриття: *атака повним перебором* і *атака по словнику*.

Поступово розроблювачі усвідомлюють необхідність застосування алгоритмів, що зарекомендували себе,, зрушуються з мертвої крапки позиції деяких країн у питанні експорту криптоалгоритмів, розробляються нові алгоритми й стандарти з більшою довжиною ключа й ефективністю для реалізації на всіх типах процесорів, від 8-бітних до RISC.

Проте, залишається величезна прірва між рівнем стійкості й надійності існуючого зараз ПЗ, що застосовує криптоалгоритми, у якому дотепер перебувають «дитячі» діри» і тим рівнем криптостійкісти, що демонструють останні, незалежно проаналізовані провідними криптоаналітиками алгоритми й протоколи, де серйозною уразливістю вважається, наприклад, та, що вимагає 2^{65} блоків шифрованого тексту й потім 2^{58} перебору варіантів або одного відкритого тексту, зашифрованого 2^{33} різними, але залежними друг від друга ключами й

потім складності аналізу, рівного 2^{57}. Хочеться сподіватися, що майбутні реалізації й застосування цих алгоритмів збережуть настільки високий ступінь їхньої надійності.

Можна виділити 4 основні групи причин ненадійності криптографічних систем: застосування нестійких алгоритмів, неправильна реалізація або застосування криптоалгоритмів, а також людський фактор. При цьому видна чітка паралель між ними й причинами порушення безпеки обчислювальних систем.

Через описані причини були або є проблеми в безпеці у всіх класів програмних продуктів, що використовують криптоалгоритми, будь то операційні системи; криптопротоколи; клієнти й сервера, їх підтримуючі; офісні програми; користувальницькі утиліти шифрування; популярні архіватори.

Для того щоб грамотно реалізувати власну криптосистему, необхідно не тільки ознайомиться з помилками інших і зрозуміти причини, по яких вони відбулися, але й, можливо, застосовувати особливі захисні прийоми програмування й спеціалізовані засоби розробки.

Класичні методи криптоаналізу

Диференціальний криптоаналіз

В 1990 році Елі Бихам і Ади Шамир увели поняття диференціального криптоаналізу. Це був новий, раніше невідомий метод криптоаналізу, що був ефективніше розкриття грубою силою.

Диференціальний криптоаналіз працює з парами шифротекстів, відкриті тексти яких містять певні відмінності. Метод аналізує еволюцію цих відмінностей у процесі проходження відкритих текстів через етапи шифрування з тим самим ключем. Просто вибираються 2 відкритих тексти з фіксованим розходженням. Можна вибрати тексти випадковим образом, аби тільки вони відрізнялися друг від друга певним чином, криптоаналітику навіть не потрібно знати їхніх значень. Потім, використовуючи розходження в що вийшли шифротекстах, привласнюються різні ймовірності різним ключам. У процесі

подальшого аналізу наступних пар шифротекстів один із ключів стане найбільш імовірним. Це і є правильний ключ.

Певні розходження пар відкритих текстів мають високу ймовірність викликати певні розходження одержуваних шифротекстів. Ці розходження називаються характеристиками. Характеристики поширюються на певну кількість етапів шифрування й по суті визначають проходження цих етапів. Існує вхідне розходження, розходження на кожному етапі й вихідне розходження – з певною ймовірністю.

Пари відкритих текстів, що відповідають характеристиці, називається правильною парою, а пари невідповідних – неправильною парою. Правильна пара підказує правильний ключ етапу (для останнього етапу характеристики), неправильна пара – випадковий ключ етапу. Щоб знайти правильний ключ етапу, потрібно просто зібрати достатня кількість припущень. Один з підключей буде зустрічатися частіше, ніж всі інші. Фактично, правильний підключ виникне із всіх випадкових можливих підключей. Інші біти повного ключа виходять за допомогою грубого взлому.

Але існує ряд проблем. По-перше, поки не буде досягнуте деяке граничне значення, імовірність успіху дуже мала. Тобто, поки не буде накопичена достатня кількість даних, виділити правильний підключ із шуму неможливо. Крім того, таке розкриття непрактично. Для зберігання ймовірностей можливих ключів необхідно використовувати лічильники, і до того ж для розкриття потрібно занадто багато даних.

Таким чином, розкриття в значній мірі теоретичне. Величезні вимоги до часу й обсягу даних, необхідних для виконання розкриття за допомогою диференціального криптоаналізу, перебувають майже поза межами досяжності. Крім того, це, насамперед розкриття з обраним відкритим текстом, воно може бути перетворене до розкриття з відомим відкритим текстом, але прийде переглянути все пари «відкритий текст – шифротекст» у пошуках корисних. Це робить розкриття ледве менш ефективним у порівнянні із грубою силою.

Криптоаналіз зі зв'язаними ключами

Криптоаналіз зі зв'язаними ключами схожий на диференціальний криптоаналіз, але він вивчає розходження між ключами. Криптоаналітик вибирає зв'язок між парою ключів, але самі ключі залишаються йому невідомі. Дані шифруються обома ключами. У варіанті з відомим відкритим текстом криптоаналітику відомі відкритий текст і шифротекст даних, шифрованими двома ключами. У варіанті з обраним відкритим текстом криптоаналітик намагається вибрати відкритий текст, зашифрований двома ключами. Це розкриття не залежить від кількості етапів шифрування, але також не реалізовано на практиці.

Лінійний криптоаналіз

Лінійний криптоаналіз являє собою інший тип криптоаналитического розкриття, винайдений Міцуру Мацуї. Розкриття використовує лінійні наближення для опису роботи блокового шифру. Це означає, що якщо виконати операцію XOR над деякими бітами відкритого тексту, потім над деякими бітами шифротекста, а потім над результатами, вийде біт, що являє собою XOR деяких біт ключа. Це називається лінійним наближенням, що може бути вірним з деякою ймовірністю p. Якщо p≠1/2, той цей зсув можна використовувати. Можна використовувати зібрані відкриті тексти й зв'язані шифротексти для припущення про значення біт ключа. Чим більше даних, тим вірніше припущення, чим більше зсув, тим швидше розкриття увінчається успіхом.

Метод зустрічі посередині

Якщо безліч ключів криптоалгоритма замкнуто щодо композиції, тобто для будь-яких ключів z_i і z_j найдеться ключ z_k такий, що результат шифрування будь-якого тексту послідовно на z_i і z_j дорівнює шифрограмі цього ж числа на z_k, тобто $F(z_j, F(z_i, x)) = F(z_k, x)$, те можна скористатися цією властивістю. Нехай нам потрібно знайти ключ z_k. Тоді для знаходження ключа z_k, необхідно знайти еквівалентну йому пари ключів z_i, z_j. Даний метод криптоаналізу заснований на «*парадоксі днів народження*». Відомо, що якщо вважати, що дні народження

розподілені рівномірно, то в групі з 24 чоловік з імовірністю 0,5 у двох чоловік дні народження збіжаться.

Нехай відомий відкритий текст x і його шифрограма y. Для тексту x будуємо базу даних, що містить випадкову безліч ключів $z^{|}$ і відповідних шифрограм $w=F(z^{|}, x)$, і впорядковуємо її по шифрограмам w. Потім підбираємо випадковим образом ключі $z^{||}$ для розшифровки тексту y і результат розшифрування $v = F(z^{|}{}^{|}, y)$ порівнюємо з базою даних. Якщо текст v виявиться рівним однієї із шифрограм w, то ключ $z^{|}\ z^{||}$ еквівалентний шуканому ключу z. Часова складність методу становить $O(\sqrt{}\ \#\ \{z\}\ log\#\{z\})$. Множник $log\#\{z\}$ ураховує складність сортування. Необхідна пам'ять дорівнює $O(\sqrt{}\ \#\ \{z\}\ log\#\{z\})$ біт або $O\sqrt{}\ \#\ \{z\})$ блоків (передбачається, що довжина блоку й довжина ключа розрізняються на обмежену константу).

Інше застосування цього методу для безлічі, що не є напівгрупою, можна продемонструвати на геш-функціях. Наприклад, для підробки підпису треба знайти два тексти, що володіють одним геш-образом. Після цього можна підписане повідомлення замінити на інше, що володіє таким же геш-образом. Пошук двох таких повідомлень можна виконати з використанням методу «зустрічі посередині». Складність пошуку дорівнює $O(\sqrt{}\ \#\{h\})$, де $\#\{h\}$ – число всіляких геш-образів. Основним недоліком даного методу є вимога до розміру пам'яті для зберігання бази даних при більших розмірах ключа.

Придатність нейромережних алгоритмів для рішення завдання криптоаналізу

У будь-якому алгоритмі шифрування завжди присутнє процедура порівняння вхідних даних з наявними в пам'яті еталонами. Поза залежністю від наявності або відсутності попередньої обробки (виділення основних ознак, перетворення в іншу форму в новому параметричному просторі й т.д.) вхід буде являти собою вектор у якому-небудь параметричному просторі, і цей вектор рівняється з векторами, які запам’ятовувані в пам'яті. Саме цю операцію й виконує більшість нейромережних моделей. Крім того, для нейромереж

розроблені потужні алгоритми навчання, що дозволяють автоматизувати процес створення й настроювання системи.

У ході проведення аналітичного огляду була вивчена предметна область, розглянуті існуючі методи криптографії й криптоаналізу, виявлені їхні основні достоїнства й недоліки. В аналітичному огляді показано, що створення системи аналізу криптографічних алгоритмів є актуальним завданням.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium,

використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4

забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні

64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки. Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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

Розробка системи криптографічного аналізу з використанням представників різних класів шифрів

Для того, щоб розробити систему криптографічного аналізу на основі використання нейрокомп'ютерної мережі, необхідно розглянути ті криптоалгоритми, які будуть піддаватися аналізу в даній магістерській роботі. До них відносяться наступні: DES, Feal, Redoc, LOKI, RC2, RC5, RSA.

Дані алгоритми обрані у зв'язку з тим, що вони є одними із самих затребуваних на сучасному етапі розвитку криптографічних засобів захисту інформації.

Шифр DES

DES являє собою блоковий шифр, він шифрує дані 64-бітовими блоками. З одного кінця алгоритму вводиться 64-бітовий блок відкритого тексту, а з іншого кінця виходить 64-бітовий блок шифротексту. DES є симетричним алгоритмом: для шифрування й дешифрування використовуються однакові алгоритм і ключ (за винятком невеликих розходжень у використанні ключа). довжина ключа дорівнює 56 бітам. (ключ звичайно представляється 64-бітовим числом, але кожний восьмий біт використовується для перевірки парності й ігнорується, біти парності є найменшими значущими бітами байтів ключа). Ключ, що може бути будь-яким 56-бітовим числом, можна змінити в будь-який момент часу. Ряд чисел вважаються слабкими ключами, але їх можна легко уникнути. Безпека повністю визначається ключем. На найпростішому рівні алгоритм не представляє нічого більшого, ніж комбінація двох основних методів шифрування: зсуву й дифузії. Фундаментальним будівельним блоком DES є застосування до тексту одиничної

комбінації цих методів (підстановка, а за нею – перестановка), що залежить від ключа. Такий блок називається етапом. DES складається з 16 етапів, однакова комбінація методів застосовується до відкритого тексту 16 разів:

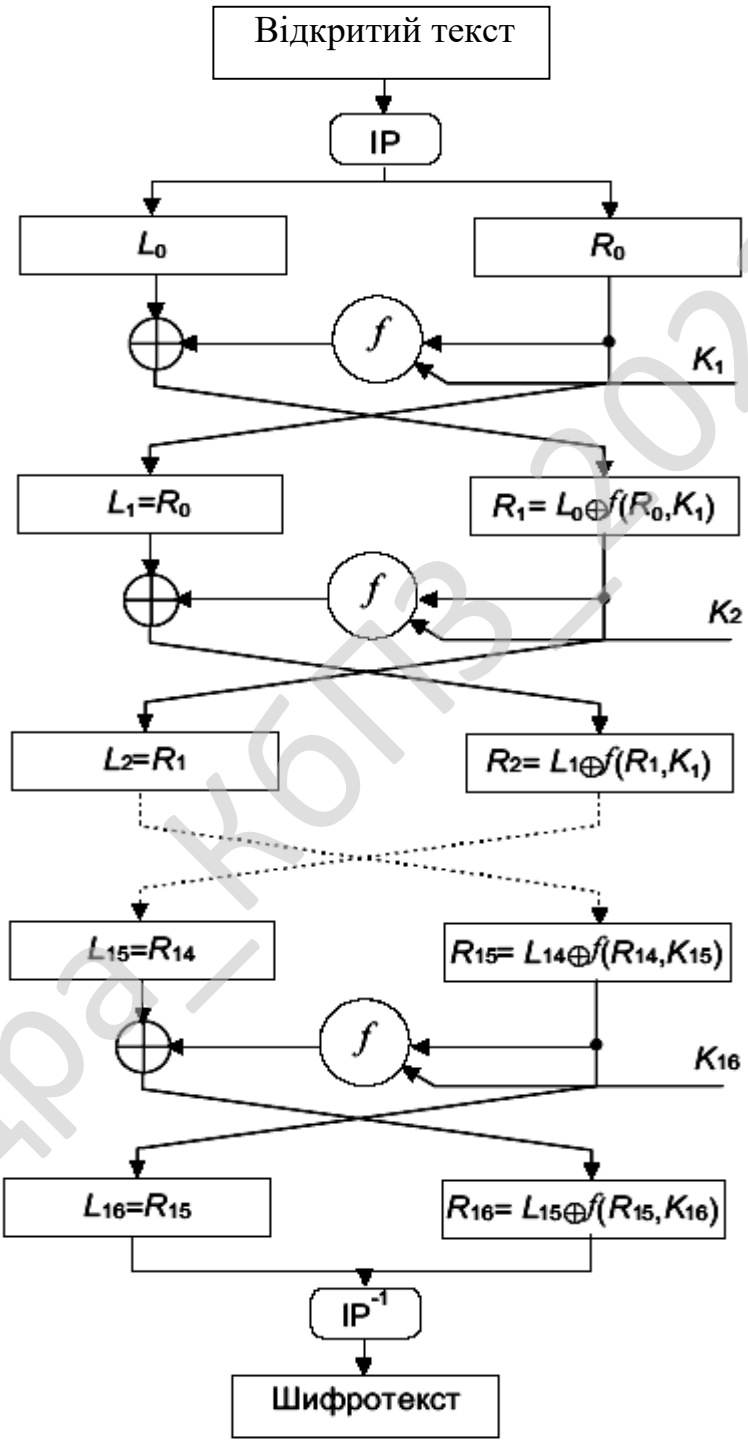

Рисунок 3.1 – Алгоритм DES

Розглянемо більш докладно алгоритм шифрування. DES працює з 64-бітовим блоком відкритого тексту. Після первісної перестановки блок розбивається на праву й ліву половини довжиною по 32 біта. Потім виконується 16 етапів однакових дій, називаних функцією f, у яких дані поєднуються із ключем. Після шістнадцятого етапу права й ліва половини поєднуються й алгоритм завершується заключною перестановкою (зворотною відносно первісної). На кожному етапі біти ключа зрушуються, і потім з 56 біт ключа вибираються 48 біт. Права половина даних збільшується до 48 біт за допомогою перестановки з розширенням, поєднується за допомогою XOR с 48 бітами зміщеного й переставленого ключа, проходить через 8 S-блоків, утворюючи 32 нових біта, і переставляється знову. Ці чотири операції й виконуються функцією f. Потім результат функції f поєднується з лівою половиною за допомогою іншого XOR. У підсумку цих дій з'являється нова права половина, а стара права половина стає нової лівою. Ці дії повторюються 16 разів, утворюючи 16 етапів DES

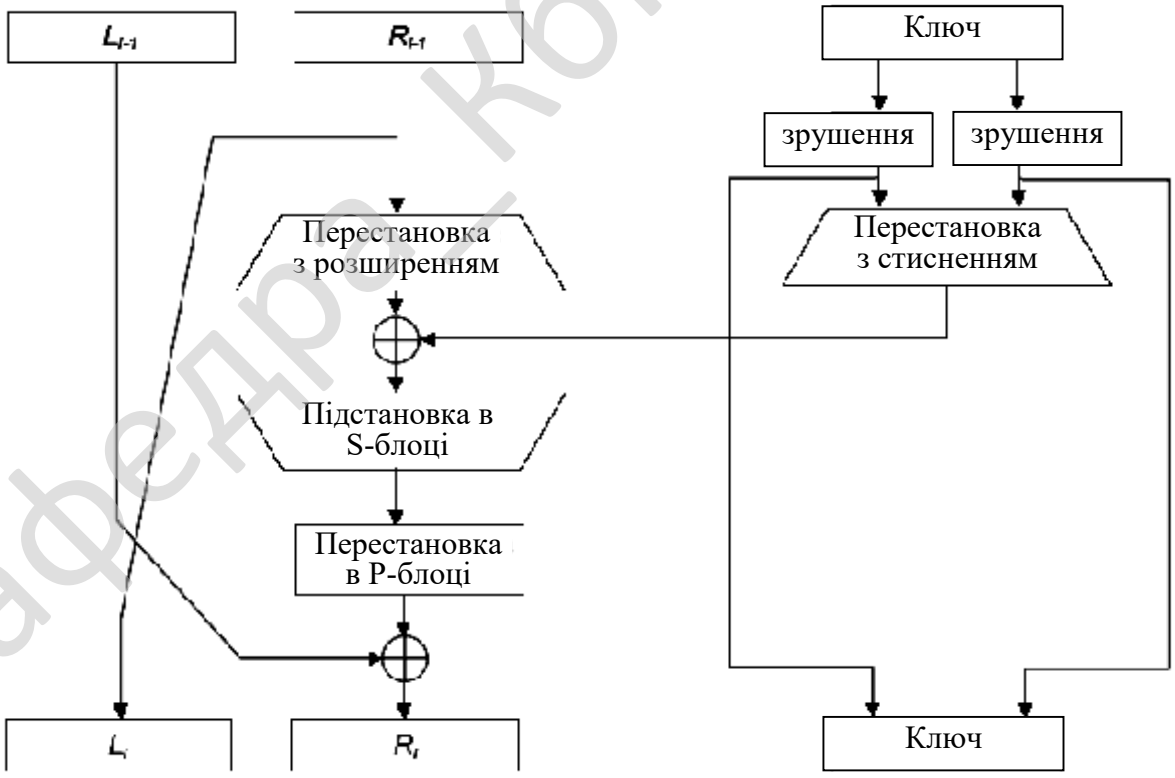

Рисунок 3.2 – Один етап DES

Якщо B_i – це результат і-ой ітерації. L_i, і R_i, – ліва й права половини B_i, K_i – 48-бітовий ключ для етапу i, а f – це функція, що виконують всі підстановки, перестановки й XOR з ключем, то етап можна представити як: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$.

Початкова перестановка виконується ще до етапу 1, при цьому вхідний блок переставляється.

Таблиця 3.1 – Початкова перестановка шифру DES

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Початкова перестановка й відповідна заключна перестановка не впливають на безпеку DES. Так як програмна реалізація цієї багатобітної перестановки нелегка, у багатьох програмних реалізаціях DES початкова й заключна перестановки не використовуються. Хоча такий новий алгоритм не менш безпечний, чим DES, він не відповідає стандарту DES і, тому, не може називатися DES.

Перетворення ключа: 64-**б**ітовий ключ DES зменшується до 56-бітового ключа відкиданням кожного восьмого біта. Ці біти використовуються тільки для контролю парності, дозволяючи перевіряти правильність ключа. Після добування 56-бітового ключа для кожного з 16 етапів DES генерується новий 48-бітовий підключ і ці підключи, K_i, визначаються наступним чином.

Таблиця 3.2 – Перетворення ключа DES

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

По перше, 56-бітовий ключ ділиться на дві 28-бітових половинки. Потім, половинки циклічно зрушуються ліворуч на один або два біти залежно від етапу.

Таблиця 3.3 – Число біт зрушення залежно від етапу DES

Етап	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після зрушення вибирається 48 з 56 біт. Так як при цьому не тільки вибирається підмножина біт, але й змінюється їхній порядок, ця операція називається **перестановка зі стиском.** Її результатом є набір з 48 біт. Наприклад, біт зрушеного ключа в позиції 33 переміщається в позицію 35 результату, а 18-й біт зрушеного ключа відкидається.

Таблиця 3.4 – Перестановка зі стиском

14	17	11	24	1	5	3	28	15	6	21	10
23	19	11	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Через зрушення для кожного підключа використовується відмінна підмножина біт ключа. Кожний біт використовується приблизно в 14 з 16 підключей, хоча не всі біти використовуються в точності однакове число раз

Перестановка з розширенням: ця операція розширює праву половину даних, R_i, від 32 до 48 біт. Так як при цьому не просто повторюються певні біти, але й змінюється їхній порядок, ця операція називається перестановкою з розширенням, У неї дві задачі: привести розмір правої половини у відповідність із ключем для операції XOR і одержати більше довгий результат, який можна буде стиснути в ході операції підстановки. Однак головний криптографічний зміст зовсім в іншому. За рахунок впливу одного біта на дві підстановки швидше зростає залежність біт результату від біт вихідних даних. Це називається **лавинним ефектом.** DES спроектований так, щоб якнайшвидше домогтися

залежності кожного біта шифротексту від кожного біта відкритого тексту й кожного біта ключа.

Перестановка з розширенням показана на 9-й. Іноді вона називається Е-б**локом** (від expansion). для кожного 4-бітовий вхідний блоки перший і четвертий біт являють собою два біти вихідного блоку, а другий і третій біти – один біт вихідного блоку. В 7-й показано, які позиції результату відповідають яким позиціям вихідних даних. Наприклад, біт вхідного блоку в позиції 3 переміститься в позицію 4 вихідні блоки, а біт вхідного блоку в позиції 21 – у позиції 30 і 32 вихідного блоку.

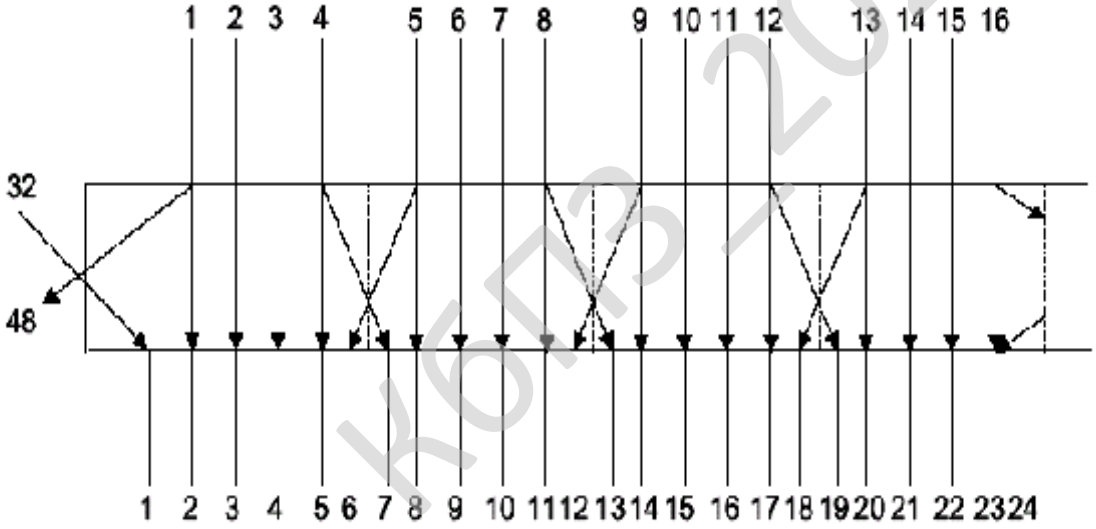

Рисунок 3.3 – Перестановка з розширенням

Хоча вихідний блок більше вхідного, кожний вхідний блок генерує унікальний вихідний блок

Таблиця 3.5 – Перестановка з розширенням DES

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

Підстановка за допомогою S-блоків: Після об'єднання стислого блоку з розширеним блоком за допомогою XOR над 48-бітовим результатом виконується операція підстановки. Підстановки виробляються у вісьмох блоках підстановки, або S-блоках (від Substitution). У кожного 8-блоку 6-бітовий вхід і 4-бітовий вихід, усього використовується вісім різних S-блоків. (для восьми S-блоків DES буде потрібно 256 байтів пам'яті.) 48 біт діляться на вісім 6-бітових підблока. Кожний окремий підблок обробляється окремим S-блоком: перший підблок – S-блоком 1, другий – S-блоком 2 і так далі.

Рисунок 3.4 – Підстановка S-блоку

Кожний **S-блок** являє собою таблицю з 2 рядків і 16 стовпців. Кожний елемент у блоці є 4– бітовим числом. По 6 вхідних бітах S-блоку визначається, під якими номерами стовпців і рядків шукати вихідне значення.

Перестановка за допомогою P-блоків: 32-бітовий вихід підстановки за допомогою S-блоків, перетасовуються відповідно до P-блоку. Ця перестановка переміщає кожний вхідний біт в іншу позицію, жоден біт не використовується двічі, і жоден біт не ігнорується. Цей процес називається прямою перестановкою або просто перестановкою. Наприклад, біт 21 переміщається в позицію 4, а біт 4 – у позицію 31.

Таблиця 3.6 – Перестановка за допомогою P-блоків

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Нарешті, результат перестановки за допомогою P-блоку поєднується за допомогою XOR. с лівою половиною первісного 64-бітового блоку. Потім ліва й права половини міняються місцями, і починається наступний етап.

Заключна перестановка є зворотною відносно початкової перестановки. Зверніть увагу, що ліва й права половини не міняються місцями після останнього етапу DES, замість цього об'єднаний блок $R_{16}L_{16}$ використовується як вхід заключної перестановки. У цьому немає нічого особливого, перестановка половинок з наступним циклічним зрушенням привела б до точно такого ж результату. Це зроблено для того, щоб алгоритм можна було використовувати як для шифрування, так і для дешифрування.

Таблиця 3.7 – Заключна перестановка DES

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Дешифрування DES

Після всіх підстановок, перестановок, операцій XOR. і циклічних зрушень можна подумати, що алгоритм дешифрування, різко відрізняючись від алгоритму шифрування, точно так заплутаний. Навпроти, різні компоненти DES були підібрані так, щоб виконувалася дуже корисна властивість: для шифрування й дешифрування використовується той самий алгоритм. DES дозволяє використовувати для шифрування або дешифрування блоку ту саму функцію. Єдина відмінність полягає в тому, що ключі повинні використовуватися у зворотному порядку. Тобто, якщо на етапах шифрування використовувалися ключі K_1, K_2, K_3, ..., K_{16}, то ключами дешифрування будуть K_{16}, K_{15}, K_{14}, ..., K_1. Алгоритм, що створює ключ для кожного етапу, також циклічний. Ключ зрушується праворуч, а число позицій зрушення дорівнює 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Шифр Feal

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних піддається операції XOR з 64 бітами ключа, а потім блок даних розщеплюється на ліву й праву половини. Об'єднання лівої й правої половин за допомогою XOR утворить нову праву половину. Ліва половина й нова права половина проходять через n етапів (спочатку чотири). На кожному етапі права половина поєднується за допомогою функції f із шістнадцятьома бітами ключа й за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після n етапів (не забувайте, що ліва й права половини не перебудовуються після n-го етапу) ліва половина знову поєднується за допомогою XOR. с правою половиною, утворюючи нову праву половину, потім ліва і права з'єднуються разом в 64-бітове ціле. Блок даних поєднується за допомогою XOR з іншими 64 бітами ключа, і алгоритм завершується.

Функція f бере 32 біта даних і 16 біт ключа й змішує їх разом. Спочатку блок даних розбивається на 8-бітові шматочки, які потім поєднуються за допомогою XOR. і заміняють один одного. Дві функції S_0 і S_1 визначаються в такий спосіб:

$S_0(a,b)$ = циклічне зрушення вліво на два біти ((a + b) mod 256)

$S_1(a,b)$ = циклічне зрушення вліво на два біти((a + b+ 1) mod 256)

Той же алгоритм може бути використаний для дешифрування. Єдиною відмінністю є те, що при дешифруванні порядок використання частин ключа міняється на зворотний. Спочатку 64-бітовий ключ ділиться на дві половини, до яких застосуються операції XOR і функції f. Два 32-бітових входи розбиваються на 8-бітові блоки, поєднувані й замінні у відповідності зі схемою. S_0 і S_1 визначаються, як показано на рисунку 3.6. Потім в алгоритмі шифрування/дешифрування використовуються 16-бітові блоки ключа.

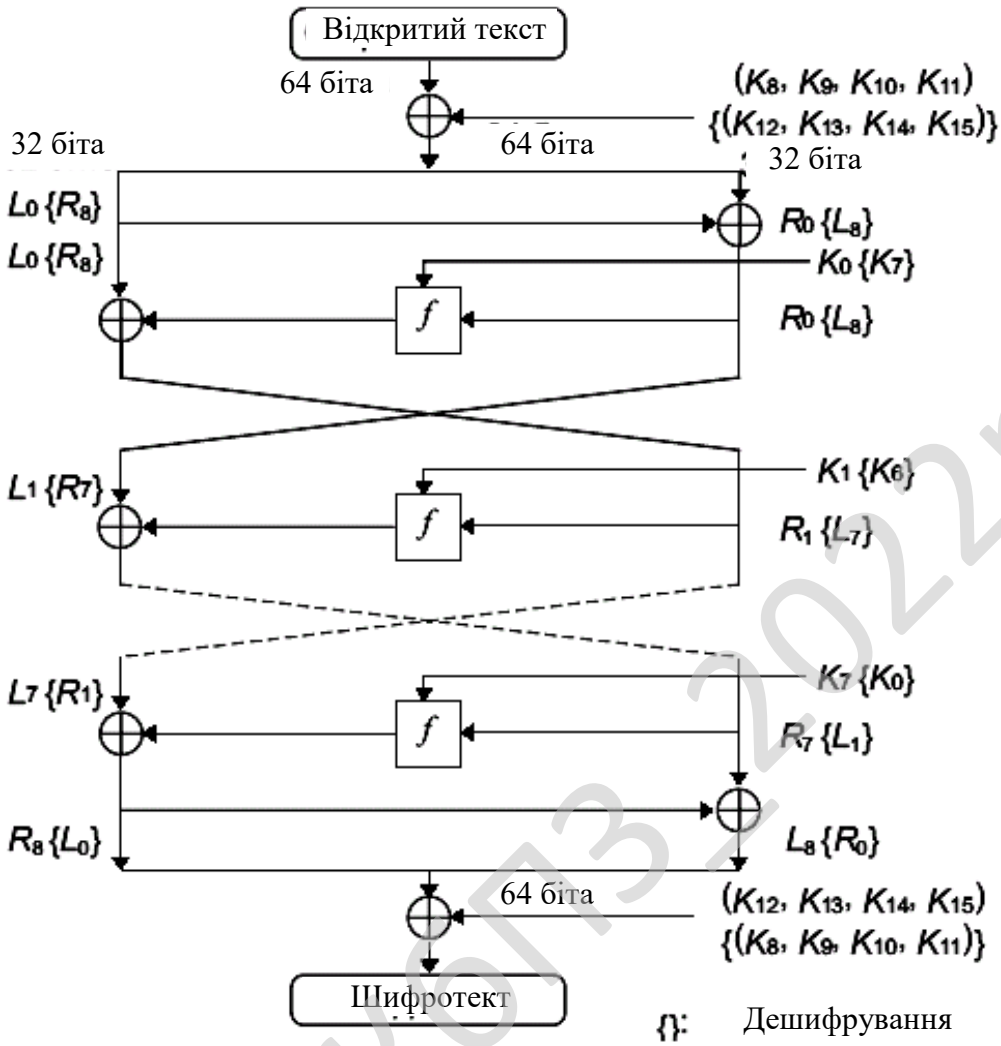

Рисунок 3.5 – Один етап Feal

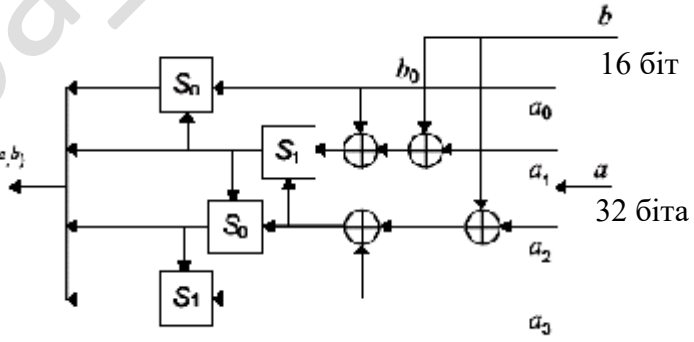

Рисунок 3.6 – Функція f

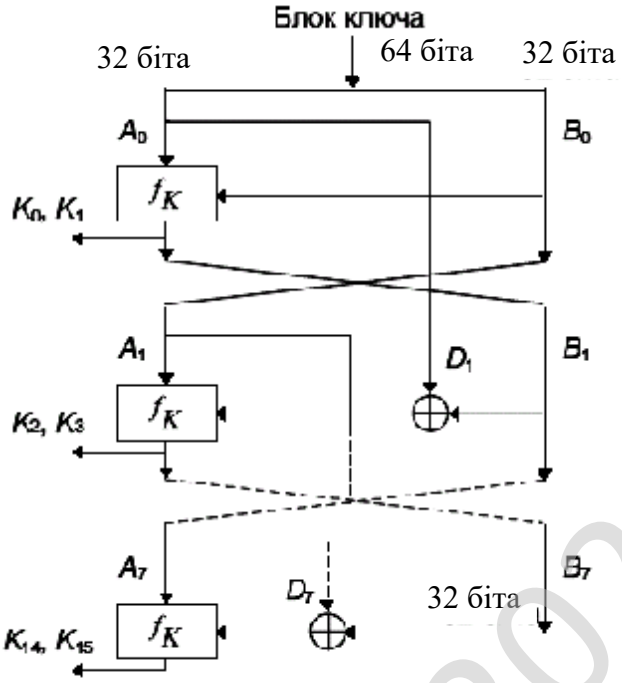

Рисунок 3.7 – Обробка ключа в Feal

Шифр REDOC

REDOC виконує всі маніпуляції – перестановки, підстановки й XOR із ключем – з байтами, цей алгоритм ефективний при програмній реалізації. REDOC використовує мінливі табличні функції. На відміну від DES, що має фіксований набір таблиць підстановок і перестановок REDOC використовує залежні від ключа й відкритого тексту набори таблиць (по суті S-блоків). В REDOC 10 етапів, кожний етап являє собою складну послідовність маніпуляцій із блоком.

Іншою унікальною особливістю є використання **масок,** які є числами, отриманими з таблиці ключів, і використовуються для вибору таблиць даної функції для даного етапу. Для вибору таблиць функції використовуються як значення даних, так і маски.

За умови, що найефективнішим засобом розкриття цього алгоритму є груба сила, REDOC дуже надійна: для розкриття ключа потрібно 2^{160} операцій. Томас Кузик (Thomas Cusick) виконав криптоаналіз одного етапу REDOC, але йому не вдалося розширити розкриття на кілька етапів. Використовуючи

диференціальний криптоаналіз, Бихам і Шамир досягли успіху в криптоаналізі одного етапу REDOC за допомогою 2300 обраних відкритих текстів Вони не змогли розширити це розкриття на кілька етапів, але їм удалося одержати три значення маски після 4 етапів.

Шифр LOKI

Механізм LOKI схожий на DES. Блок даних ділиться на ліву й праву половини й проходить через 16 етапів, що дуже схоже на DES. На кожному етапі права половина спочатку піддається операції XOR із частиною ключа, а потім над нею виконується перестановка з розширенням.

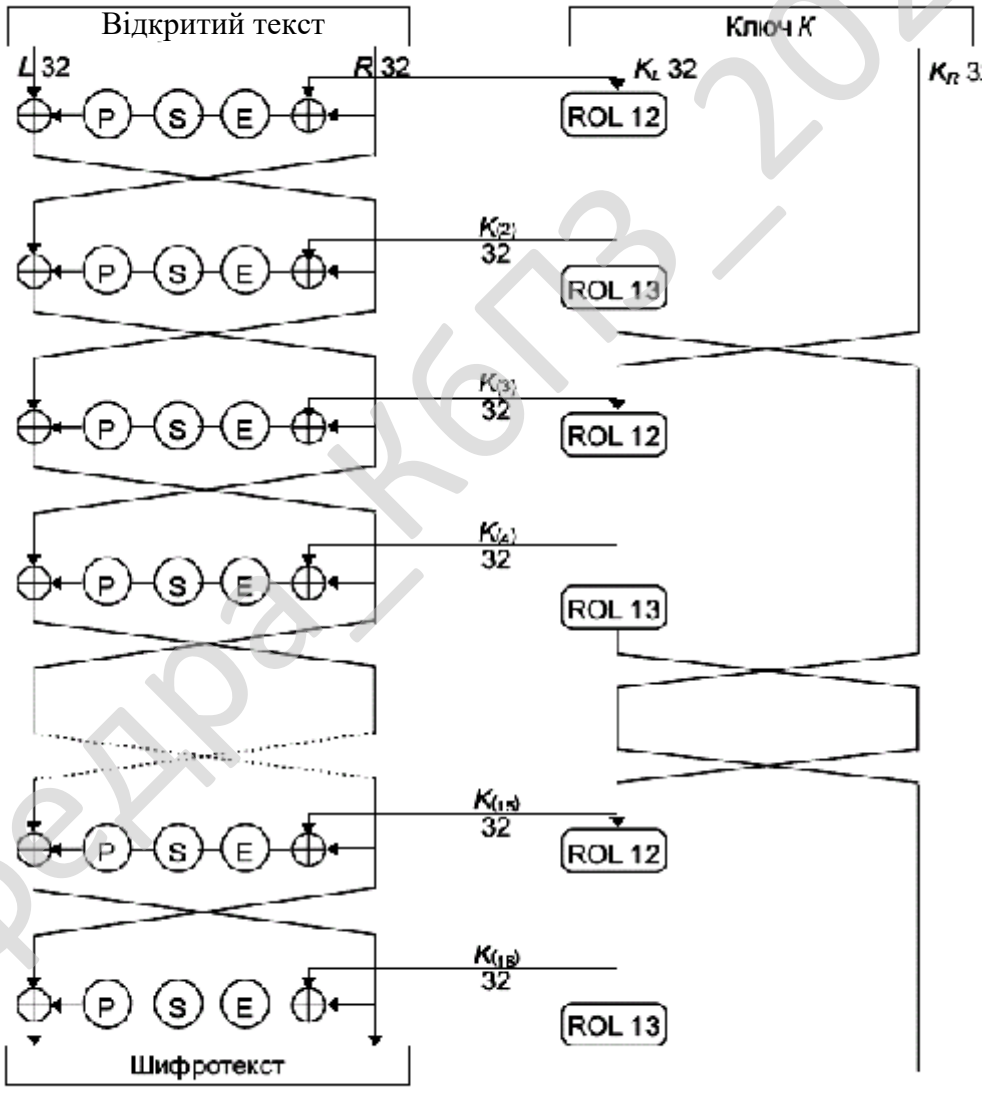

Рисунок 3.8 – Шифр LOKI91

Таблиця 3.8 – Таблиця перестановок

4	3	2	1	32	31	20	29	28	27	26	25
28	27	26	25	24	23	22	21	20	19	18	17
20	19	18	17	16	15	14	13	12	11	10	9
12	11	10	9	8	7	6	5	4	3	2	1

48-бітовий результат ділиться на чотири 12-бітових блоки, для кожного з яких виконується наступна підстановка з використанням S-блоку: береться кожний 12-бітовий вхід, по 2 крайні ліві й крайніх правих біти використовуються для одержання номера r, в 8 центральних біт утворять номер с. Результатом S-блоку – O – є наступне значення:

$$O(r,c) = (C+ ((r* 17) \oplus Oxff) \& Oxff)^{31} \bmod P_r.$$

Потім чотири 8-бітових результати знову поєднуються, утворюючи 32-бітове число, що піддається операції перестановки. Нарешті для одержання нової лівої половини виконується XOR правої половини з колишньою лівою половиною, а ліва половина стає новою правою половиною. Після16 етапів для одержання остаточного шифротексту знову виконується XOR блоку й ключа. Підключи із ключа виділяються досить прямолінійно. 64-бітовий ключ розбивається на ліву й праву половини. На кожному етапі підключем є ліва половина. Далі вона циклічно зрушується вліво на 12 або 13 біт, потім після кожних двох етапів ліва й права половини міняються місцями. Як і в DES для шифрування й дешифрування використовується той самий алгоритм із деякими змінами у використанні підключей.

Шифр RC2

RC2 являє собою алгоритм зі змінною довжиною ключа. RC2 – це шифр із 64-бітовим блоком і змінною довжиною ключа, призначений замінити DES. Відповідно до тверджень компанії програмні реалізації RC2 у три рази швидше DES. Алгоритм може використовувати ключ змінної довжини, від 0 байтів до максимальної довжини рядка, підтримуваною комп'ютерною системою, швидкість шифрування не залежить від розміру ключа. Цей ключ попередньо

використовується для заповнення 128-байтовой таблиці, що залежить від ключа. Тому множина дійсно різних ключів становить $2^{1023.}$ RC2 не використовує S-блоків, використовуються дві операції – "змішування" і "перемішування" ("mix" і "mash"), для кожного етапу вибирається одна з них.

RC2 не є ітеративним блоковим шифром. Це припускає, що RC2 більше стійкий до диференціального й лінійному криптоаналізу, чим інші блокові шифри, безпека яких опирається на копіювання схеми DES.

Шифр RC5

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

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

$A=A + S_0$

$B = B + S_0$

Для і від 1 до r:

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

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

Вихід перебуває в регістрах A и B.

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

Для i від r до 1 із кроком -1:

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

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

$B = B - S_i$

$A = A - S_0$

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

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

$S_0 = P$

Для i від 1 до $2(r + 1) - 1$:

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

де P = 0xb7e15163 і Q = 0x9e3779b9.

Нарешті, потрібно підставити L в S:

$i = j = 0$

$A = B = 0$

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

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

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

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

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

Алгоритм RSA

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

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

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

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m \pmod N$.

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

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

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

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

Опис нейрокомп’ютерної мережі

Нейрокомп’ютерна мережа зустрічного поширення

Дослідимо **нейрокомп’ютерну мережу зустрічного поширення** У процесі навчання вхідні вектори асоціюються з відповідними вихідними векторами. Ці вектори можуть бути двійковими, що складаються з нулів і одиниць, або безперервними. Коли мережа навчена, прикладання вхідного вектора приводить до необхідного вихідного вектора. Узагальнююча здатність мережі дозволяє одержувати правильний вихід навіть при додатку вхідного вектора, що є неповним або злегка невірним. Це дозволяє використовувати дану мережу для розпізнавання образів, відновлення образів і посилення сигналів.

Структура мережі

На рисунку 3.9 показана спрощена версія прямої дії мережі зустрічного поширення. На ньому ілюструються функціональні властивості цієї парадигми.

Нейрони шару 0 (показані кружками) служать лише крапками розгалуження й не виконують обчислень. Кожний нейрон шару 0 з'єднаний з кожним нейроном шару 1 (називаного шаром Кохонена) окремою вагою $w_{mn.}$ Ці ваги в цілому розглядаються як матриця ваг ***W***. Аналогічно, кожний нейрон у шарі Кохонена (шар 1) з'єднаний з кожним нейроном у шарі Гроссберга (шар 2) вагою $v_{пр}$. Ці ваги утворять матрицю ваг **V**. Все це досить нагадує інші мережі, розходження, однак, складається в операціях, виконуваних нейронами Кохонена й Гроссберга.

Як і багато інших мереж, зустрічне поширення функціонує у двох режимах: у нормальному режимі, при якому приймається вхідний вектор **X** и видається вихідний вектор **Y**, і в режимі навчання, при якому подається вхідний вектор і ваги коректуються, щоб дати необхідний вихідний вектор.

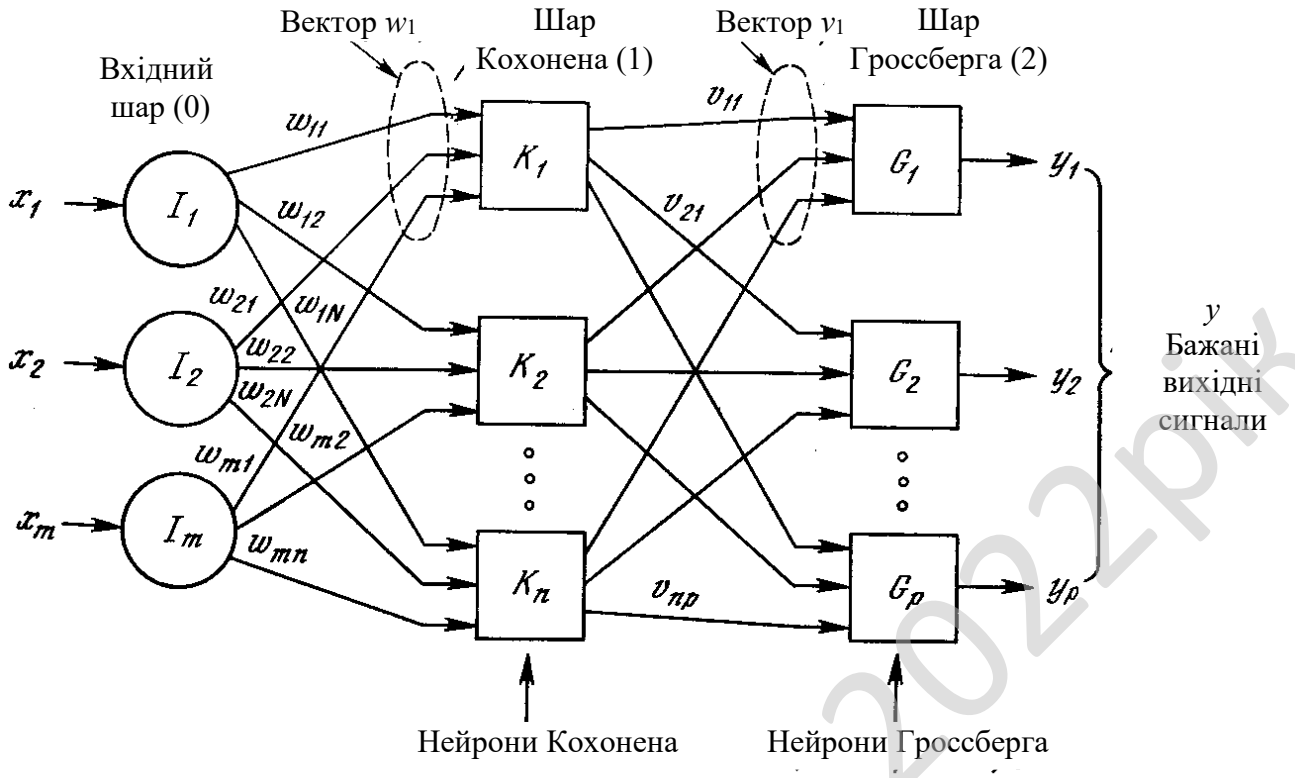

Рисунок 3.9 – Мережа із зустрічним розпізнаванням без зворотних зв'язків

Нормальне функціонування

Шари Кохонена

У своїй найпростішій формі шар Кохонена функціонує в дусі «переможець забирає все», тобто для даного вхідного вектора один і тільки один нейрон Кохонена видає на виході логічну одиницю, всі інші видають нуль. Нейрони Кохонена можна сприймати як набір електричних лампочок, так що для будь-якого вхідного вектора загоряється одна з них.

Асоційоване з кожним нейроном Кохонена множина ваг з'єднує його з кожним входом. Наприклад, на рисунку 3.9 нейрон Кохонена K_1 має ваги w_{11}, w_{21}, …, w_{m1}, які складають вектор ваги $\mathbf{W}_1$. Вони з'єднуються через вхідний шар із вхідними сигналами x_1, x_2, …, x_m, , які складають вхідний вектор X. Подібно нейронам більшості мереж вихід NET кожного нейрона Кохонена є просто сумою зважених входів. Це може бути виражене в такий спосіб:

$$NET_j = w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{mj}x_m \qquad (3.1)$$

де NET_j – це вихід NET нейрона Кохонена j,

$$\mathrm{NET_j} = \sum_i x_i w_{ij} \tag{3.2}$$

або у векторному запису

$$\mathrm{N = XW}, \tag{3.3}$$

де N – вектор виходів NET шару Кохонена.

Нейрон Кохонена з максимальним значенням NET є «переможцем». Його вихід дорівнює одиниці, в інших він дорівнює нулю.

Шар Гроссберга

Шар Гроссберга функціонує в подібній манері. Його вихід NET є зваженою сумою виходів $k_1, k_2, ..., k_n$ шару Кохонена, що утворюють вектор **К.** Вектор з'єднуючих ваг, позначений через **V**, складається з ваг $v_{11}, v_{21}, ..., v_{np}$. Тоді вихід NET кожного нейрона Гроссберга є

$$\mathrm{NET_j} = \sum_i k_i w_{ij}, \tag{3.4}$$

де $\mathrm{NET_j}$ – вихід j-го нейрона Гроссберга, або у векторній формі

$$\mathrm{Y = KV}, \tag{3.5}$$

де Y – вихідний вектор шару Гроссберга,

К – вихідний вектор шару Кохонена,

V – матриця ваг шару Гроссберга.

Якщо шар Кохонена функціонує таким чином, що лише в одного нейрона величина NET дорівнює одиниці, а в інших дорівнює нулю, то лише один елемент вектора К відмінний від нуля, і обчислення дуже прості. Фактично кожний нейрон шару Гроссберга лише видає величину ваги, що зв'язує цей нейрон з єдиним ненульовим нейроном Кохонена.

Навчання шару Кохонена

Шар Кохонена класифікує вхідні вектори в групи схожих. Це досягається за допомогою такого підстроювання ваг шару Кохонена, що близькі вхідні вектори активують той самий нейрон даного шару. Потім задачею шару Гроссберга є одержання необхідних виходів.

Навчання Кохонена є самонавчанням, що протікає без учителя. Тому важко (і не потрібно) пророкувати, який саме нейрон Кохонена буде

активуватися для заданого вхідного вектора. Необхідно лише гарантувати, щоб у результаті навчання розділялися несхожі вхідні вектори.

Попередня обробка вхідних векторів

Досить бажано (хоча й не обов'язково) нормалізувати вхідні вектори перед тим, як пред'являти їхньої мережі. Це виконується за допомогою ділення кожного компонента вхідного вектора на довжину вектора. Ця довжина перебуває добуванням квадратного кореня із суми квадратів компонент вектора. В алгебраїчному записі

$$x'_{\mathrm{i}} = \frac{x_{\mathrm{i}}}{\sqrt{x_1^2 + x_2^2 + \ldots + x_{\mathrm{n}}^2}} \qquad (3.6)$$

Це перетворює вхідний вектор в одиничний вектор з тим же самим напрямком, тобто у вектор одиничної довжини в n-мірному просторі.

Рівняння (3.6) узагальнює добре відомий випадок двох вимірів, коли довжина вектора дорівнює гіпотенузі прямокутного трикутника, утвореного його *х* и *в* компонентами, як це треба з відомої теореми Піфагора. На рисунку 3.10 такий двовимірний вектор V представлений у координатах *х-у,* причому координата *х* дорівнює чотирьом, а координата *у* – трьом. Квадратний корінь із суми квадратів цих компонентів дорівнює п'яти. Ділення кожного компонента V на п'ять дає вектор V з компонентами 4/5 і 3/5, де V' указує в тому же напрямку, що й V, але має одиничну довжину.

На рисунку 3.11 показано кілька одиничних векторів. Вони кінчаються в крапках одиничної окружності (окружності одиничного радіуса), що має місце, коли в мережі лише два входи. У випадку трьох входів вектори представлялися б стрілками, що кінчаються на поверхні одиничної сфери. Ці подання можуть бути перенесені на мережі, що мають довільне число входів, де кожний вхідний вектор є стрілкою, що кінчається на поверхні одиничної гіперсфери (корисною абстракцією, хоча й не безпосередньої візуалізації, що допускає).

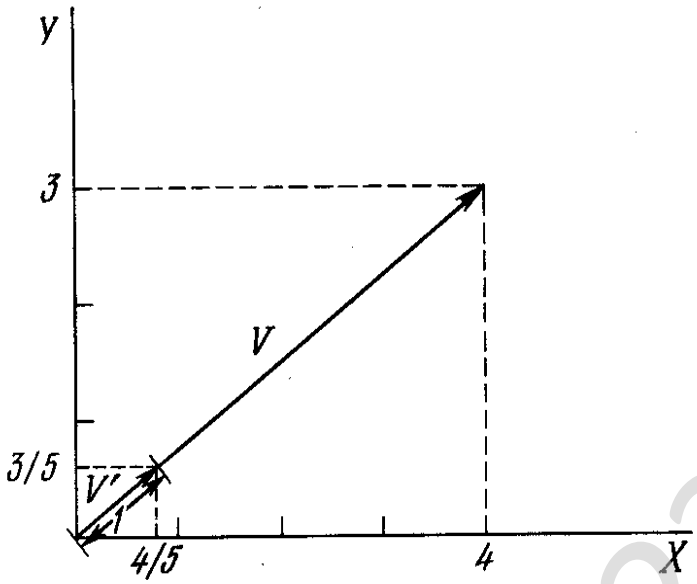

Рисунок 3.10 – Одиничний вхідний вектор

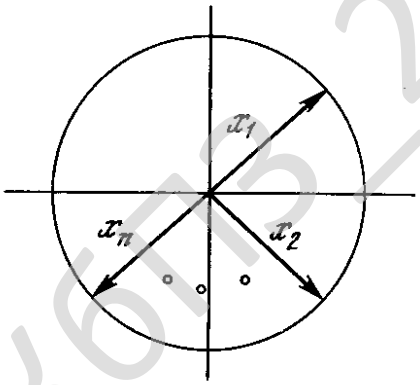

Рисунок 3.11 – Двовимірні одиничні вектори на одиничній окружності

При навчанні шару Кохонена на вхід подається вхідний вектор і обчислюються його скалярні добутки з векторами ваг, зв'язаними з усіма нейронами Кохонена. Нейрон з максимальним значенням скалярного добутку оголошується «переможцем» і його ваги підбудовуються. Так як скалярний добуток, використовуваний для обчислення величин NET, є мірою подібності між вхідним вектором і вектором ваг, то процес навчання складається у виборі нейрона Кохонена з ваговим вектором, найбільш близьким до вхідного вектора, і подальшому наближенні вагового вектора до вхідного. Знову відзначимо, що процес є самонавчанням, виконуваним без учителя. Мережа самоорганізується

таким чином, що даний нейрон Кохонена має максимальний вихід для даного вхідного вектора. Рівняння, що описує процес навчання має такий вигляд:

$$w_н = w_c + \alpha(x - w_c), \quad (3.7)$$

де $w_н$ – нове значення ваги, що з'єднує вхідний компонент x з нейроном, що виграв;

w_c – попереднє значення цієї ваги;

α – коефіцієнт швидкості навчання, що може варіюватися в процесі навчання.

Кожна вага, пов'язана з нейроном, що виграв, Кохонена, змінюється пропорційно різниці між його величиною й величиною входу, до якого він приєднаний. Напрямок зміни мінімізує різниця між вагою і його входом.

На рисунку 3.12 цей процес показаний геометрично у двовимірному виді. Спочатку знаходиться вектор $X - W_c$, для цього проводиться відрізок з кінця W у кінець X. Потім цей вектор коротшає множенням його на скалярну величину α, меншу одиниці, у результаті чого виходить вектор зміни δ. Остаточно новий ваговий вектор $W_н$ є відрізком, спрямованим з початку координат у кінець вектора δ. Звідси можна бачити, що ефект навчання складається в обертанні вагового вектора в напрямку вхідного вектора без істотної зміни його довжини.

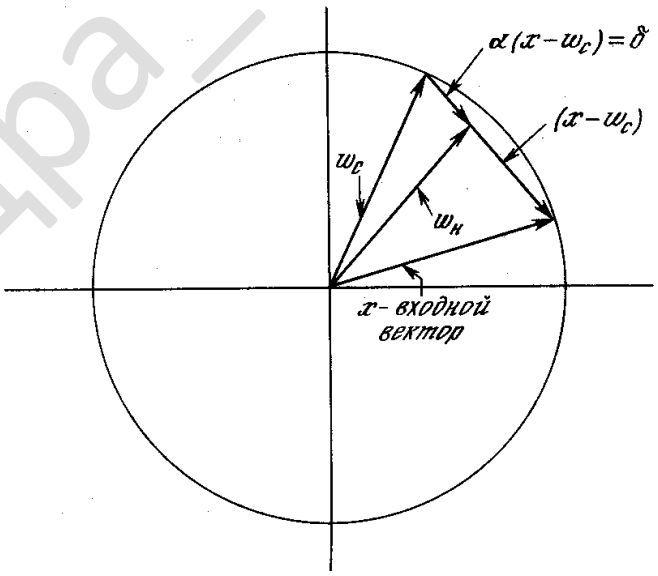

Рисунок 3.12 – Обертання вагового вектора в процесі навчання

$W_н$ – вектор нових вагових коефіцієнтів,

$W_с$ – вектор старих вагових коефіцієнтів

Змінна *к* є коефіцієнтом швидкості навчання, що спочатку звичайно дорівнює ~ 0,7 і може поступово зменшуватися в процесі навчання. Це дозволяє робити більші початкові кроки для швидкого грубого навчання й менші кроки при підході до остаточної величини.

Якби з кожним нейроном Кохонена асоціювався один вхідний вектор, то шар Кохонена міг би бути навчений за допомогою одного обчислення на вагу. Ваги нейрона-переможця прирівнювалися б до компонентів навчального вектора ($\alpha = 1$). Як правило, множина, що навчає, включає багато подібних між собою вхідних векторів, і мережа повинна бути навчена активувати один й той самий нейрон Кохонена для кожного з них. У цьому випадку ваги цього нейрона повинні виходити усередненням вхідних векторів, які повинні його активувати. Поступове зменшення величини (зменшує вплив кожного навчального кроку, так що остаточне значення буде середньою величиною від вхідних векторів, на яких відбувається навчання. Таким чином, ваги, асоційовані з нейроном, приймуть значення поблизу «центра» вхідних векторів, для яких даний нейрон є «переможцем».

Вибір початкових значень вагових векторів

Всім вагам мережі перед початком навчання варто надати початкові значення. Загальноприйнятою практикою при роботі з нейронними мережами є присвоювання вагам невеликих випадкових значень. При навчанні шару Кохонена випадково обрані вагарні вектори варто нормалізувати. Остаточні значення вагових векторів після навчання збігаються з нормалізованими вхідними векторами. Тому нормалізація перед початком навчання наближає вагові вектори до їхніх остаточних значень, скорочуючи, таким чином, що навчає процес.

Рандомизація ваг шару Кохонена може породити серйозні проблеми при навчанні, так як в результаті її вагові вектори розподіляються рівномірно по

поверхні гіперсфери. Через те, що вхідні вектори, як правило, розподілені нерівномірно й мають тенденцію групуватися на відносно малій частині поверхні гіперсфери, більшість вагових векторів будуть так вилучені від будь-якого вхідного вектора, що вони ніколи не будуть давати найкращої відповідності. Ці нейрони Кохонена будуть завжди мати нульовий вихід і виявляться марними. Більше того, ваг, що залишилися, які дають найкращі відповідності, може виявитися занадто мало, щоб розділити вхідні вектори на класи, які розташований близько друг до друга на поверхні гіперсфери.

Допустимо, що є кілька множин вхідних векторів, всі множини подібні, але повинні бути розділені на різні класи. Мережа повинна бути навчена активувати окремий нейрон Кохонена для кожного класу. Якщо початкова щільність вагових векторів в околиці навчальних векторів занадто мала, то може виявитися неможливим розділити подібні класи через те, що не буде достатньої кількості вагових векторів в околиці, яка має для нас інтерес, щоб приписати по одному з них кожному класу вхідних векторів.

Навпаки, якщо кілька вхідних векторів отримані незначними змінами з того самого зразка й повинні бути об'єднані в один клас, то вони повинні включати той самий нейрон Кохонена. Якщо ж щільність вагових векторів дуже висока поблизу групи злегка різних вхідних векторів, то кожний вхідний вектор може активувати окремий нейрон Кохонена. Це не є катастрофою, так як шар Гроссберга може відобразити різні нейрони Кохонена в той самий вихід, але це марнотратна витрата нейронів Кохонена.

Найбільш бажане рішення полягає в тому, щоб розподіляти вагові вектори відповідно до щільності вхідних векторів, які повинні бути розділені, поміщаючи тим самим більше вагових векторів в околиці великої кількості вхідних векторів. На практиці це нездійсненно, однак існує кілька методів наближеного досягнення тих же цілей.

Одне з рішень, відоме за назвою *методу опуклої комбінації* (convex combination method), полягає в тому, що всі ваги прирівнюються до однієї й тієї же величині

$$w_i = \frac{1}{\sqrt{n}}, \tag{3.8}$$

де *n* – число входів і, отже, число компонентів кожного вагового вектора.

Завдяки цьому всі вагові вектори збігаються й мають одиничну довжину. Кожному ж компоненту входу **X** надається значення

$$x_i = \alpha x_i + \frac{1-\alpha}{\sqrt{n}}, \tag{3.9}$$

де *n* – число входів.

На початку α дуже мало, внаслідок чого всі вхідні вектори мають довжину, близьку до $1/\sqrt{n}$, і майже збігаються з векторами ваг. У процесі навчання мережі α поступово зростає, наближаючись до одиниці. Це дозволяє розділяти вхідні вектори й остаточно приписує їм їхні справжні значення. Вагові вектори відслідковують одну або невелику група вхідних векторів і наприкінці навчання дають необхідну картину виходів. Метод опуклої комбінації добре працює, але сповільнює процес навчання, так як вагові вектори підбудовуються до мети, що змінюється. Інший підхід складається в додаванні шуму до вхідних векторів. Тим самим вони піддаються випадковим змінам, схоплюючи зрештою ваговий вектор. Цей метод також працездатний, але ще більше медленен, чим метод опуклої комбінації.

Третій метод починає з випадкових ваг, але на початковій стадії навчального процесу підбудовує всі ваги, а не тільки пов'язані з нейроном Кохонена, що виграв. Тим самим вагові вектори переміщаються ближче до області вхідних векторів. У процесі навчання корекція ваг починає вироблятися лише для найближчих до переможця нейронів Кохонена. Цей радіус корекції поступово зменшується, так що зрештою коректуються тільки ваги, пов'язані з нейроном Кохонена, що виграв.

Ще один метод наділяє кожний нейрон Кохонена «Почуттям справедливості». Якщо він стає переможцем частіше своєї законної частки часу (приблизно $1/k$, де k – число нейронів Кохонена), він тимчасово збільшує свій поріг, що зменшує його шанси на виграш, даючи тим самим можливість навчатися й іншим нейронам.

У багатьох додатках точність результату істотно залежить від розподілу ваг. На жаль, ефективність різних рішень вичерпним образом не оцінена й залишається проблемою.

Розробка нейрокомп'ютерной мережі в застосуванні до криптоаналізу

При рішенні задачі криптоаналізу ми вибрали варіант у якому на етапі навчання в нас є наступні вхідні дані: відкритий текст, ключ, алгоритм шифрування, закритий текст.

На етапі криптоаналізу в нас є тільки закритий текст.

На етапі розпізнавання відбувається виділення із криптотекста знайомих системі зразків і подання їхнім одним нейроном або нейронним ансамблем на наступних рівнях. Як при навчанні, так і при розпізнаванні вхідні вектора є нечіткими, тобто є невеликий розкид векторів, що належать до одного класу. У зв'язку із цим нейромережа, що здійснює цю операцію, повинна мати певну здатність до статистичного усереднення. Навпроти, може виявитися, що група векторів перебуває в безпосередній близькості друг до друга, але всі вони представляють різні класи. Тоді нейромережа повинна визначати тонкі розходження між векторами. Ще одна вимога до нейромережі низького рівня обробки сигналу – навчання без учителя, тобто здатність самостійно розділяти вхідні сигнали на класи.

Велика кількість нейромережних алгоритмів виконують функцію поділу вхідного криптотекста на класи.

Відомі 3 математичні моделі цього поділу:

1. Поділ вхідних даних гіперплощинами (простий персептрон).

Застосування цього алгоритму виправдано тільки для задач, що володіють високою лінійністю. Наприклад, можна побудувати нейромережу, що розбиває крапки (0,0) і (1,1) на два класи для двовимірного сигналу, але неможливо вирішити задачу по розбивці крапок (0,0), (1,1) – перший клас, і (0,1), (1,0) – другий. Це широко відомий приклад нездатності простого персептрона вирішити задачу «або що виключає»

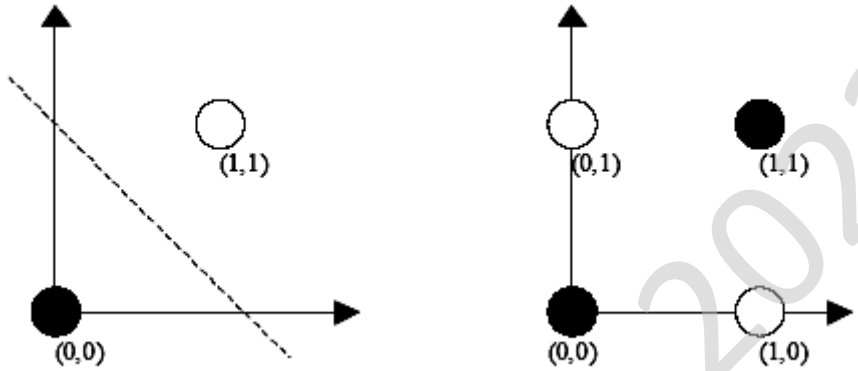

Рисунок 3.13 – Теорема Мінського

2. Поділ вхідних даних гіперповерхнями (багатошарові персептрони).

При послідовному з'єднанні шарів, подібних простому персептрону, з'являється можливість комбінувати гіперплощини й одержувати гіперповерхні досить складної форми, у тому числі й замкнуті. Така нейромережа у принципі при достатнім числі нейронів здатна розділяти сигнали на класи практично будь-якої складності. Але застосування таких нейромереж обмежене складністю їхнього навчання. Був розроблений потужний алгоритм, називаний «алгоритмом зворотного поширення помилки», але й він вимагає значного часу навчання й не гарантує мінімального значення помилки (небезпека влучення в локальні мінімуми).

3. Пошук найбільшої відповідності (найменшого кутового або лінійного стану). При нормалізованих векторах входу, усі вектора розташовуються на поверхні гіперсфери.

Існує модель нейромережі, що відповідає цим вимогам – це мережа зустрічного поширення. В оригіналі вона являє собою об'єднання двох гарно відомих алгоритмів: карти Кохонена, що самоорганізується, й шару Гроссберга. У процесі навчання вхідні вектори асоціюються з відповідними вихідними векторами. Коли мережа навчена, додаток вхідного вектора приводить до необхідного вихідного вектора. Узагальнююча здатність мережі дозволяє одержувати правильний вихід навіть при додатку вхідного вектора, що є неповним або злегка невірним.

Схематично мережа зустрічного напрямку зображена на рисунку 3.14.

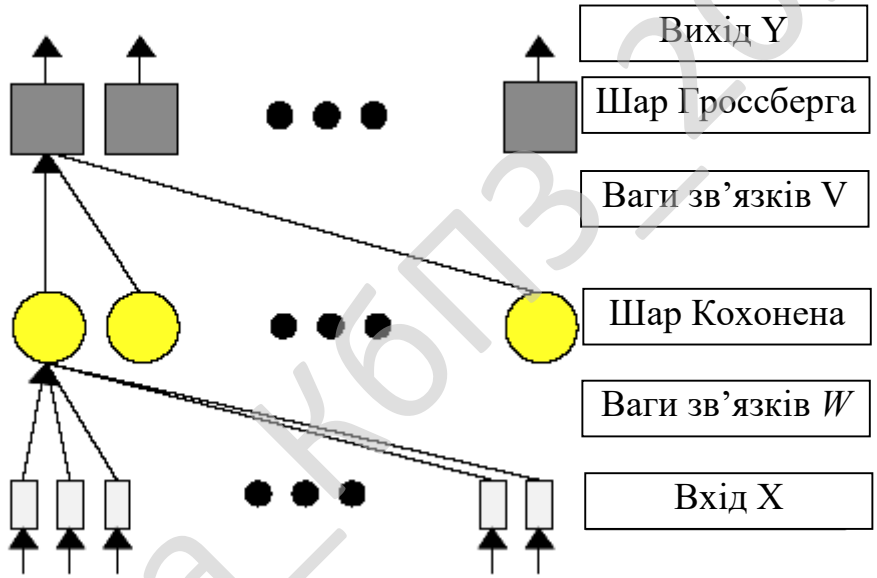

Рисунок 3.14 – Мережа зустрічного поширення

Поширення даних у такій мережі відбувається в такий спосіб: вхідний вектор нормується на 1.0 і подається на вхід, що розподіляє його далі через матрицю ваг *W*. Кожний нейрон у шарі Кохонена обчислює суму на своєму вході й залежно від стану навколишніх нейронів цього шару стають активні або неактивним (рівні 1.0 і 0.0). Нейрони цього шару функціонують за принципом конкуренції, тобто в результаті певної кількості ітерацій активним залишається один нейрон або невелика група, «пухирець активності». Цей механізм

називається латеральним гальмуванням і докладно розглянутий у багатьох джерелах. Так як відпрацьовування цього механізму вимагає значних обчислювальних ресурсів, у даній моделі він замінений знаходженням нейрона з максимальною активністю й присвоєнням йому активності 1.0, а всім іншим нейронам 0.0. Таким чином, спрацьовує нейрон, для якого вектор входу ближче всього до вектора ваг зв'язків.

Якщо мережа перебуває в режимі навчання, то для нейрона, що виграв, відбувається корекція ваг матриці зв'язку за формулою

$$w_н = w_c + \alpha(x - w_c), \quad (3.9)$$

де $w_н$ – нове значення ваги,

w_c – старе значення,

α – швидкість навчання,

x – величина входу.

Геометрично це правило ілюструє наступний рисунок.

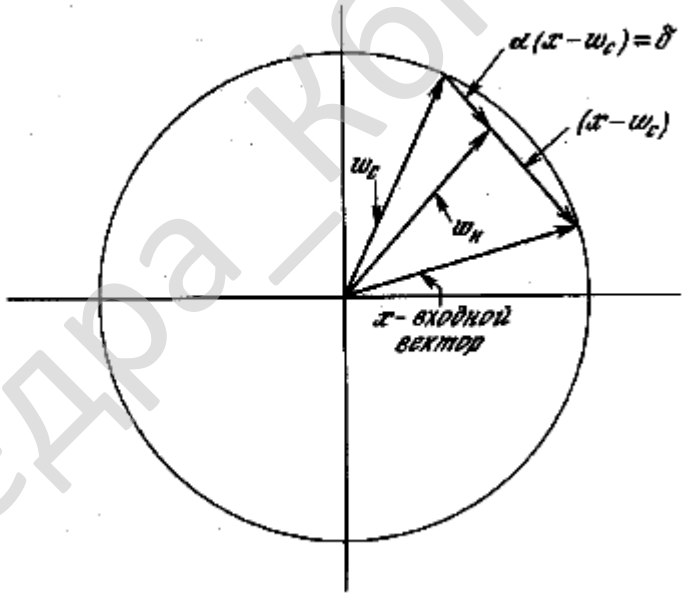

Рисунок 3.15 – Корекція ваг нейрона Кохонена

Так як вхідний вектор x нормований, тобто розташований на гіперсфері одиничного радіуса в просторі ваг, то при корекції ваг за цим правилом

відбувається поворот вектора ваг убік закритого тексту. Поступове зменшення швидкості повороту α дозволяє зробити статистичне усереднення вхідних векторів, на які реагує даний нейрон. Проблема: вибір початкових значень ваг. Так як наприкінці навчання вектора ваг будуть розташовуватися на одиничній окружності, то на початку їх також бажано віднормувати на 1.0. У розглянутій нами моделі вектора ваг вибираються випадковим образом на окружності одиничного радіуса.

Проблема: якщо ваговий вектор виявиться далеко від області входу, він ніколи не дасть найкращої відповідності, завжди буде мати нульовий вихід, отже, не буде коректуватися й виявиться марним. Нейронів, що залишилися, може не вистачити для поділу вхідного простору на класи. Для рішення цієї проблеми пропонується багато алгоритмів, тут же застосовується правило «бажання працювати»: якщо який або нейрон довго не перебуває в активному стані, він підвищує ваги зв'язків доти, поки не стане активним і не почне піддаватися навчанню. Цей метод дозволяє також вирішити проблему тонкої класифікації: якщо утвориться група вхідних даних, розташованих близько друг до друга, із цією групою асоціюється й велика кількість нейронів Кохонена, які розбивають її на класи. Правило «бажання працювати» записується в наступній формі:

$$w_н = w_c + w_c\beta\,(1 - a), \qquad (3.10)$$

де $w_н$ – нове значення ваги,

w_c – старе значення,

β – швидкість модифікації,

a – активність нейрона. Чим менше активність нейрона, тим більше збільшуються ваги зв'язків.

Далі сигнал через матрицю ваг V надходить на шар Гроссберга тут шар спрацьовує по старому методу.

Алгоритм навчання

Вхідні дані: навчальна вибірка (набір вхідних векторів).

Вихідні дані: скоректовані зв'язки.

1. Пред'явити мережі вхідний вектор.

2. Виконувати ітерації до встановлення стабільного стану.

3. Для всіх вузлів мережі виконати корекцію зв'язків згідно (2) або (3).

4. Повторювати [1-3] для кожного вхідного вектора.

Розроблена система криптоаналізу є досить універсальною, не вимоглива до пам'яті й показала себе ефективніше, ніж класичні методи криптоаналізу. Достоїнствами даної системи є: швидкість аналізу, легкість адаптації, універсальність (існуючі алгоритми не захищені від такого виду аналізу).

3.2 Розробка структурної схеми

Система аналізу криптографічних алгоритмів складається із двох підсистем: підсистеми криптографічного захисту інформації з використанням представників різних класів шифрів і підсистеми криптоаналізу. Структурна схема системи аналізу криптографічних алгоритмів представлена на рисунку 3.16.

Рисунок 3.16 – Структурна схема системи криптоаналізу

З нього ми бачимо, що існують чотири структурні блоки які взаємодіють між собою:

– вхідні данні;

– криптоалгоритми, які потребують криптоаналізу;

– власне сама система криптоаналізу, яка складається з використання нейромережі на основі шарів Кохонена та Гроссберга;

– відкритий текст, у якому даються дані про можливість криптоаналізу, того або іншого криптоалгоритму.

Користувач вибирає файл який буде вхідним текстом та відкриває його у програмі. Потім обирає криптоалгоритм, який хоче протестувати.

Файл, обраний користувачем, шифрується вибраним крптоалгоритмом. Зашифрований файл підлягає криптоаналізу на базі нейронної мережі та на основі його результатів визначається стійкійкість вибраного алгоритму шифрування.

3.3 Розробка функціональної схеми

Розглянемо функціональну схему розробленої системи. Вона зображена на рисунку 3.17.

Як видно з рисунку система складається з наступних елементів:

– Зашифрованого тексту.

– Нейронної мережі зустрічного розподілу.

– Блоку порівняння вхідних даних з образами відомими системі.

– Блоку навчаючої вибірки.

– Відкритого тексту.

Спочатку на вхід системи подається навчаюча вибірка, за допомогою якої відбувається навчання нейронної мережі. Потім в програмі відкривається зашифрований текст. Нейронна мережа намагається розпізнати алгоритм

шифрування та розшифрувати його без ключа, ще відбувається за допомогою порівняння вхідних даних з образами відомими системі після навчання.

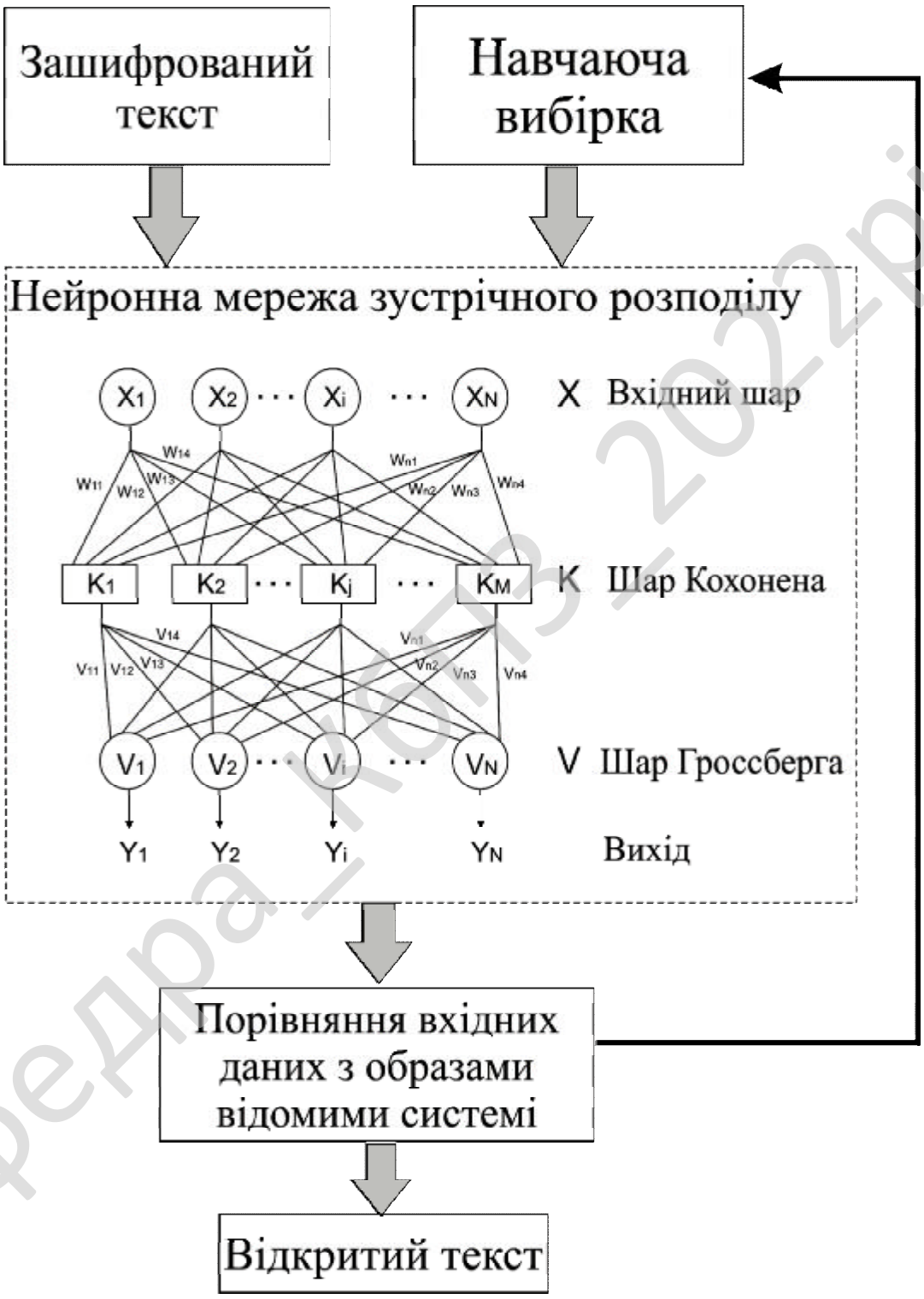

Рисунок 3.17 – Функціональна схема системи криптоаналізу

Після розшифрування тексту відбувається додавання нової інформації до бази знань. З кожним наступним використанням програма має все більше інформації для дешифрування.

У якості системи криптоаналізу використана нейронна мережа зустрічного розподілу, що має три шари:

1. Вхідний шар, X.
2. Шар Кохонена, K.
3. Шар Гроссберга, V.

Архітектура зустрічного розподілу вдало поєднує у собі переваги можливості узагальнення інформації мережі Кохонена й простоту навчання вихідної зірки Гроссберга. Ця архітектура ідеально підходить для швидкого моделювання систем на початкових етапах досліджень із подальшим переходом, якщо це буде потрібно, на значно більш дорожчий, але більш точний метод навчання зі зворотним поширенням помилок.

Нейронна мережа зустрічного розподілу навчається на вибірці пар векторів (X,Y) задачі подання відображення XY. Чудовою особливістю цієї мережі є здатність навчанню також і відображенню сукупності XY у себе. При цьому, завдяки узагальненню, з'являється можливість відновлення пари (XY) по одному відомому компоненту (X або Y). При пред'явлені на етапі розпізнавання тільки вектора X (з нульовим початковим Y) виконується пряме відображення – відновлюється Y, і навпаки, при відомому Y може бути відновлений відповідний йому X. Можливість рішення як прямої, так і зворотної задачі, а також гібридної задачі по відновленню окремих відсутніх компонентів робить дану нейромережну архітектуру унікальним інструментом.

Мережа зустрічного розподілу складається із двох шарів нейронів: шару Кохонена та шару Гроссберга. У режимі функціонування (розпізнавання) нейрони шару Кохонена працюють за принципом "переможець-забирає-все", визначаючи кластер, до якого належить вхідний образ. Потім вихідна зірка шару Гроссберга по сигналу нейрона-переможця в шарі Кохонена відтворює на виходах мережі

відповідний образ.

Навчання ваг шару Кохонена виконується без учителя на основі самоорганізації. Вхідний вектор спочатку нормується, зберігаючи напрямок. Після виконання однієї ітерації навчання визначається нейрон переможець, стан його порушення встановлюється рівним одиниці, і тепер можуть бути модифіковані ваги відповідної йому зірки Гроссберга. Темпи навчання нейронів Кохонена й Гроссберга повинні бути погоджені . У шарі Кохонена навчаються ваги всіх нейронів в околиці переможця, що поступово звужується до одного нейрона.

Навчена нейронна мережа зустрічного розподілу може функціонувати й у режимі інтерполяції, коли в шарі Кохонена залишається не один, а декілька переможців. Тоді рівні їхньої активності пропорційно нормуються, щоб у сумі становити одиницю, а вихідний вектор визначається по сумі вихідних векторів кожної з активних зірок Гроссберга. У такий спосіб нейронна мережа робить лінійну інтерполяцію між значеннями вихідних векторів, що відповідають декільком кластерам.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.18.

З рисунку видно, що на початку роботи програми відбувається процес навчання нейронної мережі. Потім користувач завантажує ваговий файл та файл з криптотекстом. Криптотекст порівнюється з образами відомими системі, та на основі цього процесу відбувається криптоаналіз.

По закінченню дешифрування криптотексту, на екран виводиться відкритий розшифрований текст. До бази знань додається новий образ. Потім обчислюються показники стійкості застосованого до тексту алгоритму шифрування та виводяться на екран.

Потім користувач може вийти з програми, або перевірити на стійкість інший криптоалгоритм.

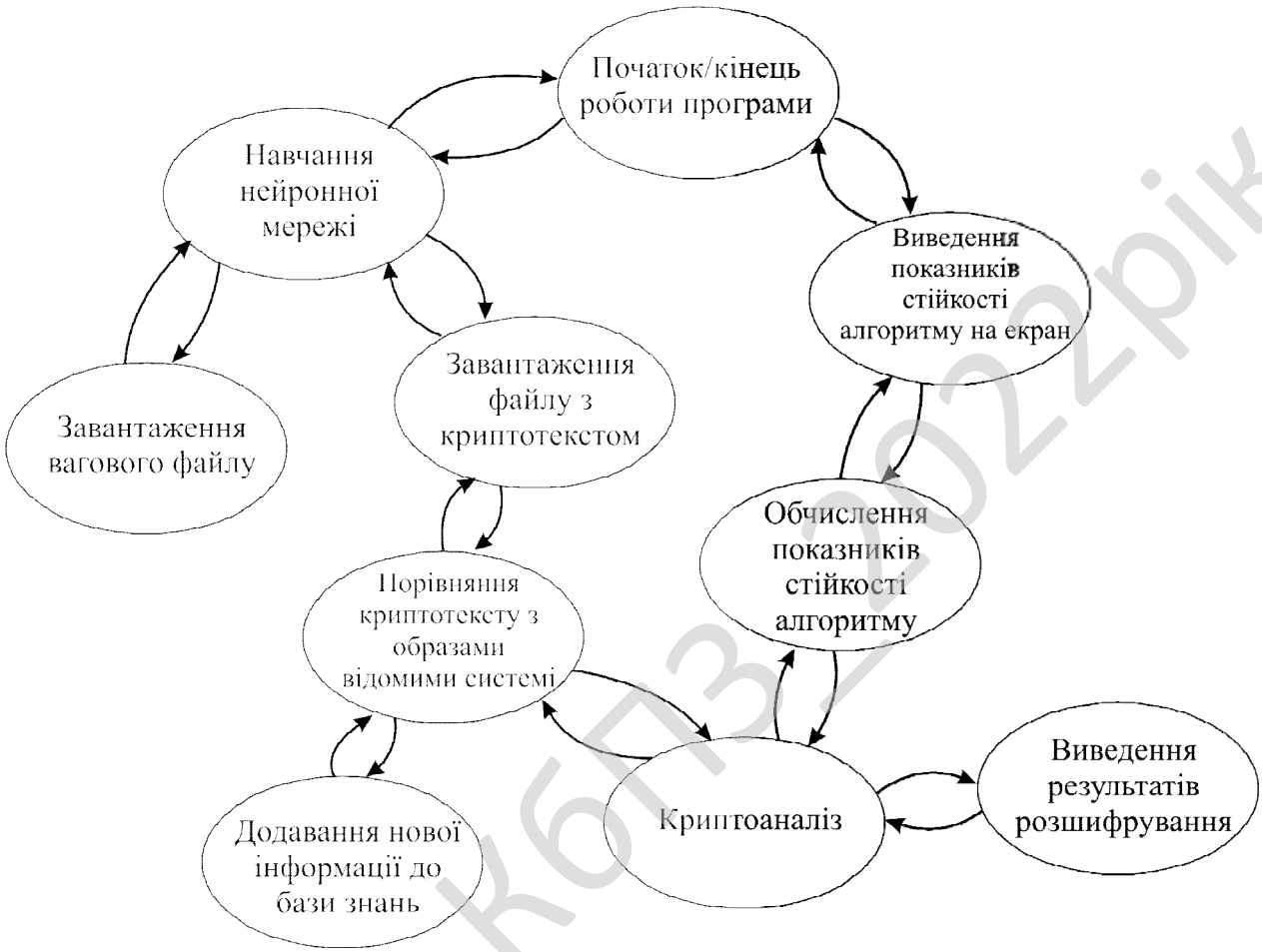

Рисунок 3.18 – Діаграма процесів розробленої системи

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

Блок-схема алгоритму роботи основної програми зображена на рисунку 4.1. Після запуску програми на екран виводиться головне вікно програми. Потім починається навчання нейронної мережі. При бажанні користувача, він може завантажити файли з ваговими коефіцієнтами для нейронної мережі. Далі слід відкрити у програмі файл з криптотекстом. Зміст відкритого файлу з'явиться у толі програми з відповідною назвою. Якщо користувач впевнився, що відкрив заме той файл, і програма не видала повідомлення про помилку формату, то він може запустити процес криптоаналізу.

Криптоаналіз здійснює нейронна мережа зустрічного розподілу, порівнюючи відомі їй образи з завантаженим файлом. Після того як нейронній мережі вдається розшифрувати криптотекст, відкритий текст виводиться у відповідне поле програми.

Потім програма обчислює показники стійкості застосованого алгоритму шифрування та виводить їх значення на екран. Також на екран виводиться назва алгоритму, яким було зашифровано криптотекст. Після цього користувач може вийти з програми, або запустити обробку іншого файлу.

Розглянемо процес навчання нейронної мережі. Нейронна мережа навчається за допомогою деякого процесу, що модифікує її ваги. Якщо навчання успішне, то пред'явлення мережі множини вхідних сигналів приводить до появи бажаної множини вихідних сигналів.

Є два класи навчальних методів: детерміністський та стохастичний. У даній роботі використаний стохастичний метод.

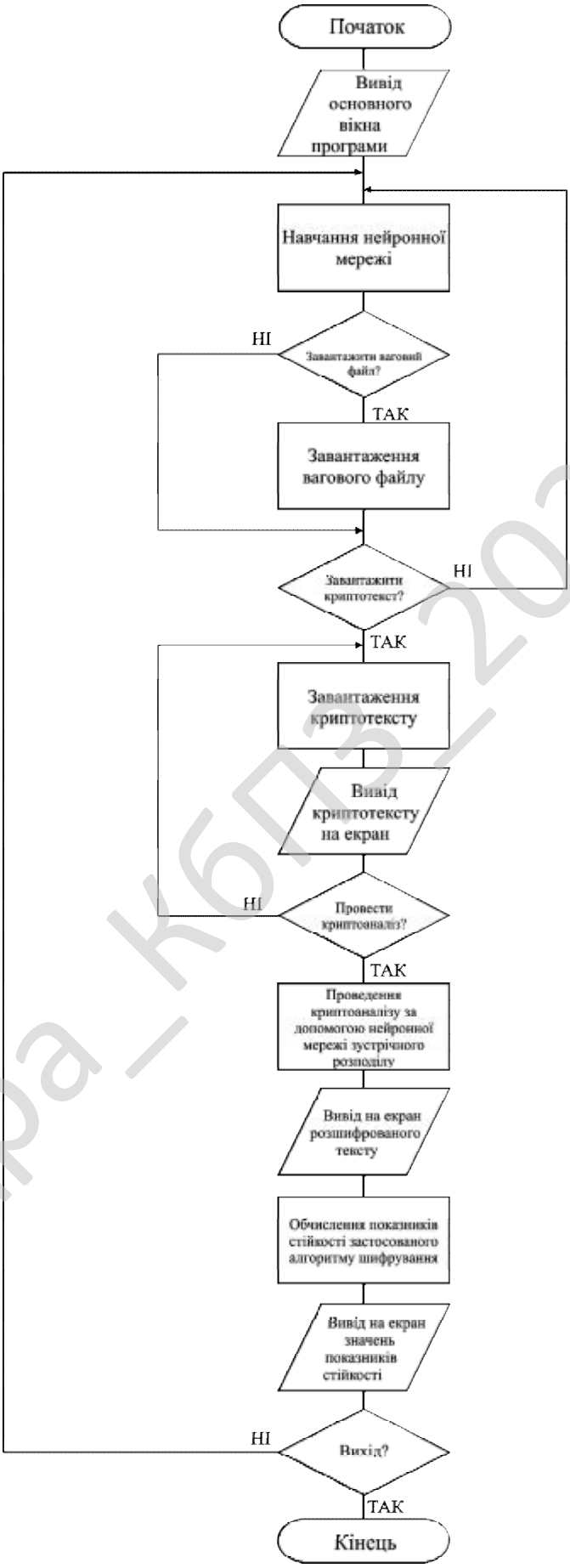

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

Детерміністський метод навчання крок за кроком здійснює процедуру корекції ваг мережі, засновану на використанні їхніх поточних значень, а також величин входів, фактичних виходів і бажаних виходів. *Стохастичні методи навчання* виконують псевдовипадкові зміни величин ваг, зберігаючи ті зміни, які ведуть до поліпшень. Вихід нейрона є тут зваженою сумою його входів, що, перетворена за допомогою нелінійної функції.

Для навчання мережі використана наступна підпрограма:

1. Вибрати вагу випадковим чином і підкоректувати її на невелике випадкове значення. Пред'явити множину входів і обчислити виходи, що отримуються.

2. Порівняти ці виходи з бажаними виходами й обчислити величину різниці між ними. Загальноприйнятий метод полягає в знаходженні різниці між фактичним і бажаним виходами для кожного елементу навчальної пари, зведення різниць у квадрат і знаходження суми цих квадратів. Метою навчання є мінімізація цієї різниці, що часто називається *цільовою функцією.*

3. Вибрати вагу випадковим чином і підкоректувати її на невелике випадкове значення. Якщо корекція допомагає (зменшує цільову функцію), то зберегти її, у протилежному випадку повернутися до первісного значення ваги.

4. Повторювати кроки з 1 по 3, поки мережа не буде навчена в достатньому ступені.

Блок-схема алгоритму роботи даної підпрограми наведена на рисунку 4.2.

Даний алгоритм має на меті мінімізувати цільову функцію, але може потрапити, як у пастку, у невдале рішення. На рисунку 4.3 показано, як це може мати місце в системі з єдиною вагою.

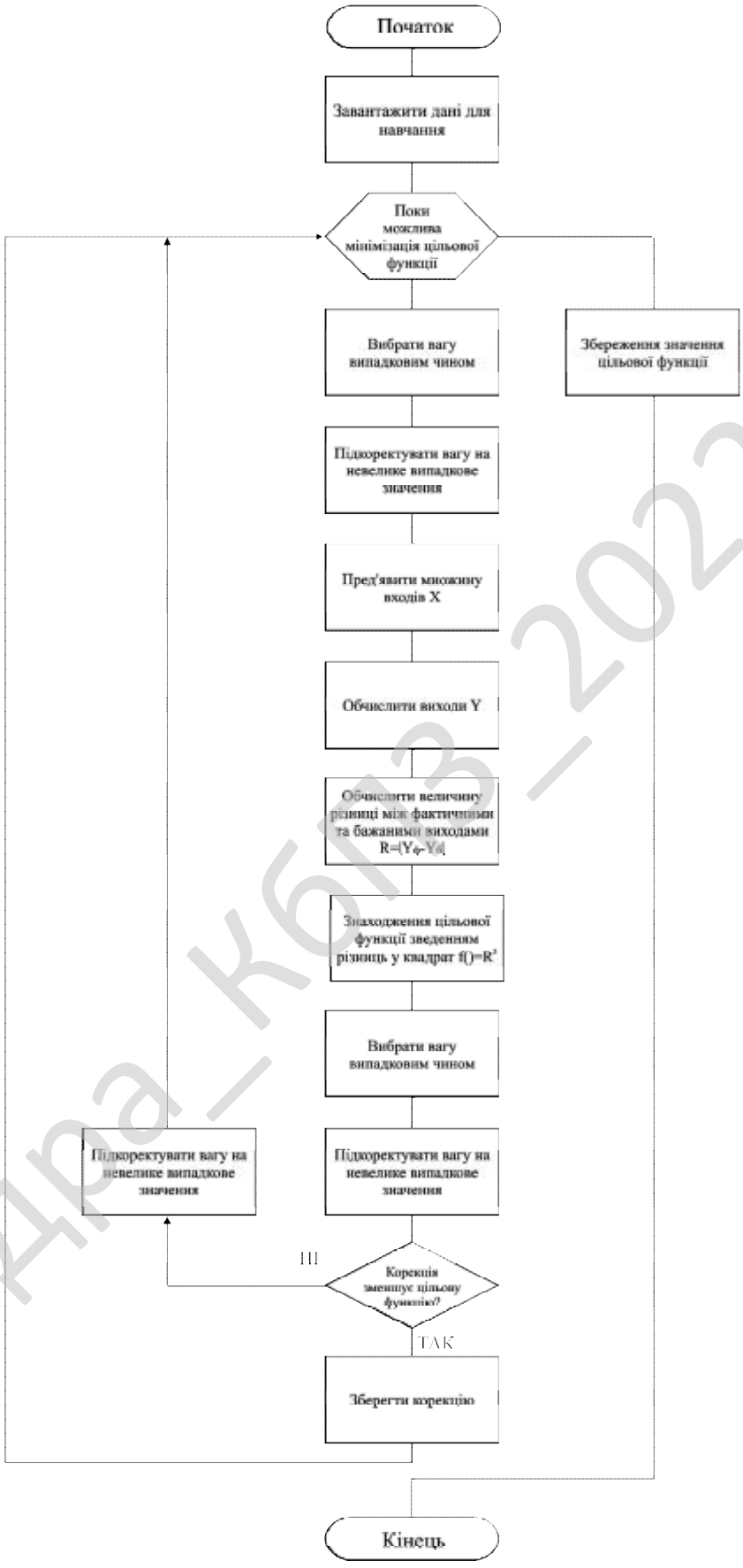

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми навчання нейронної мережі

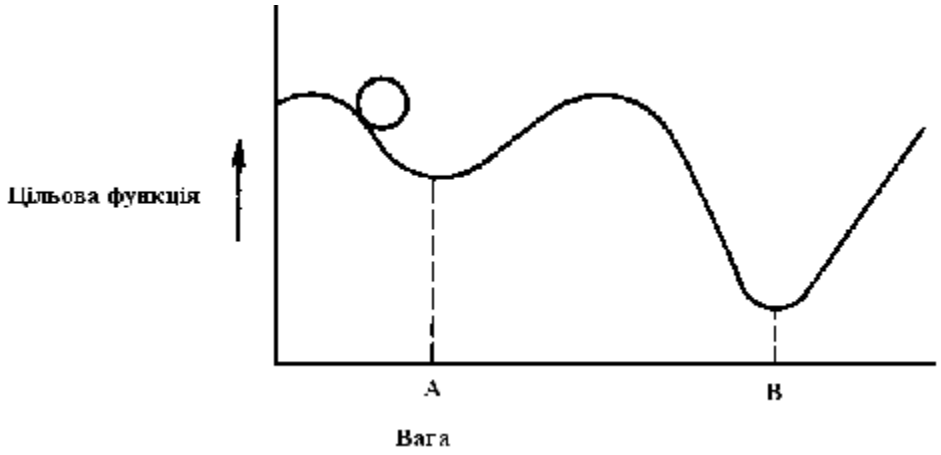

Рисунок 4.3 – Проблема локальних мінімумів

Допустимо, що спочатку вага взята рівним значенню в точці А. Якщо випадкові кроки по вазі малі, то будь-які відхилення від точки А збільшують цільову функцію й будуть відкинуті. Краще значення ваги, прийняте в точці В, ніколи не буде знайдене, і система буде піймана в пастку локальним мінімумом, замість глобального мінімуму в точці В. Якщо ж випадкові корекції ваги дуже великі, то як точка А, так і точка В будуть часто відвідуватися, але те ж саме буде мати місце й для кожної іншої точки. Вага буде мінятися так різко, що він ніколи не встановиться в бажаному мінімумі.

Корисна стратегія для запобігання подібних проблем складається в більших початкових кроках і поступовому зменшенні розміру середнього випадкового кроку. Це дозволяє мережі вириватися з локальних мінімумів і в той же час гарантує остаточну стабілізацію мережі.

З локальних мінімумів досаждають всім алгоритмам навчання, заснованим на пошуку мінімуму, включаючи персептрон і мережі зворотного поширення, і представляють серйозну й широко розповсюджені труднощі, які часто не помічаються. Стохастичні методи дозволяють вирішити цю проблему. Стратегія корекції ваг, що змушує ваги приймати значення глобального оптимуму в точці В, можлива.

Припустимо, що на рисунку 4.2 зображена кулька на поверхні в коробці. Якщо коробку сильно потрясти в горизонтальному напрямку, то кулька буде

швидко перекочуватися від одного краю до іншого. Ніде не затримуючись, у кожний момент кулька буде з рівною ймовірністю перебувати в будь-якій точці поверхні.

Якщо поступово зменшувати силу струшування, то буде досягнута умова, при якій кулька буде на короткий час «застрявати» у точці В. При ще більш слабкому струшуванні кулька буде на короткий час зупинятися як у точці А, так і в точці В. При безперервному зменшенні сили струшування буде досягнута критична точка, коли сила струшування достатня для переміщення кульки із точки А в точку В, але недостатня для того, щоб кулька могла видрати з У в А. Таким чином, остаточно кулька зупиниться в точці глобального мінімуму, коли амплітуда струшування зменшиться до нуля.

Нейронні мережі можуть навчатися по суті тим же самим чином за допомогою випадкової корекції ваг. Спочатку робляться більші випадкові корекції зі збереженням тільки тих змін ваг, які зменшують цільову функцію. Потім середній розмір кроку поступово зменшується, і глобальний мінімум зрештою досягається.

Розглянемо алгоритм криптоаналізу, на основі якого нейронна мережа розшифровує криптотекст. Для криптоаналізу був використаний метод Полларда.

Цей імовірнісний метод заснований на наступному факті. Якщо на деякій кінцевій множині M визначити випадкове відображення f і застосувати його по черзі до всіх елементів M, а ребрами відповідності – $y=f(x)$ для $x,y \subseteq M$. Оскільки множина M скінченна, то цей граф повинен містити дерева, коріння яких з'єднані в цикли. Для випадкового відображення f довжина циклу дорівнює $O(\sqrt{\#M})$ і висота дерева в середньому дорівнює $O(\sqrt{\#M})$.

Для знаходження ключа, наприклад у криптоалгоритмі, заснованому на задачі логарифму на деякій групі, досить вирішити задачу про зустріч на графі випадкового відображення. Для цього із двох різних стартових точок x_0', x_0'' будуються траєкторія до входу в цикл. Потім одна із двох кінцевих точок, що лежать у циклі, фіксується, а траєкторія іншої триває до зустрічі з фіксованою

точкою. Для функції f і точки входу x_0 довжина траєкторії становить O(√ #M) кроків. Типовий вид цієї траєкторії містить граничний цикл довжини O(√ #M) і відрізок до входу в цикл приблизно такої ж довжини. Як індикатор замикання траєкторії Поллард запропонував використовувати рівність $x_i = x_{2i}$, де x_i – *i*-та точка траєкторії для входу x_0. Ця рівність буде виконуватися завжди. Значення індексу *i* не перевищує суми довжини шляху до входу в цикл.

У середньому складність знаходження рівності $x_i = x_{2i}$ дорівнює $3\sqrt{(p/8)}\#M$. Складність зустрічі, коли обидві точки лежать у циклі, дорівнює $0{,}5\sqrt{(p/8)}\#M$. Таким чином, підсумкова складність дорівнює $6{,}5\sqrt{(p/8)}\#M$.

Цей метод дозволяє відмовитися від використання великого обсягу пам'яті в порівнянні з методом зустрічі посередині. Його часова складність менша на множник $O(\log\#M)$. Складність цього методу становить $O(\sqrt{\#M})$ кроків і вимагає пам'яті обсягу $O(1)$ блоків.

Розглянемо як ілюстрація методу Полларда алгоритм знаходження колізії (двох аргументів, що дають однакове значення геш-функції) для обчислювальної моделі з обсягом пам'яті $O(v)$. Такими аргументами будуть елементи множини M, стрілки від яких під дією геш-функції f ведуть у точку входу в цикл. Практично алгоритм зводиться до знаходження точки входу в цикл. Алгоритм (рисунок 4.4):

1. Увійти в цикл, використовуючи рівність $x_i = x_{2i} = t$.

2. Виміряти довжину циклу m, застосовуючи послідовно відображення f до елемента t до одержання рівності $f^m(t)=t$.

3. Розбити цикл m на v інтервалів однакової або близької довжини й створити базу даних, запам'ятавши й упорядкувавши початкові точки інтервалів.

4. Для стартової вершини п.1 виконувати одиночні кроки до зустрічі з якою-небудь точкою з бази даних п.3. Відзначити початкову й кінцеву точки інтервалу, на якому відбулася зустріч.

5. Стерти попередню й створити нову базу даних з v точок, розбивши інтервал, на якому відбулася зустріч, на рівні по довжині частини, запам'ятавши й відсортувавши початкові точки інтервалів.

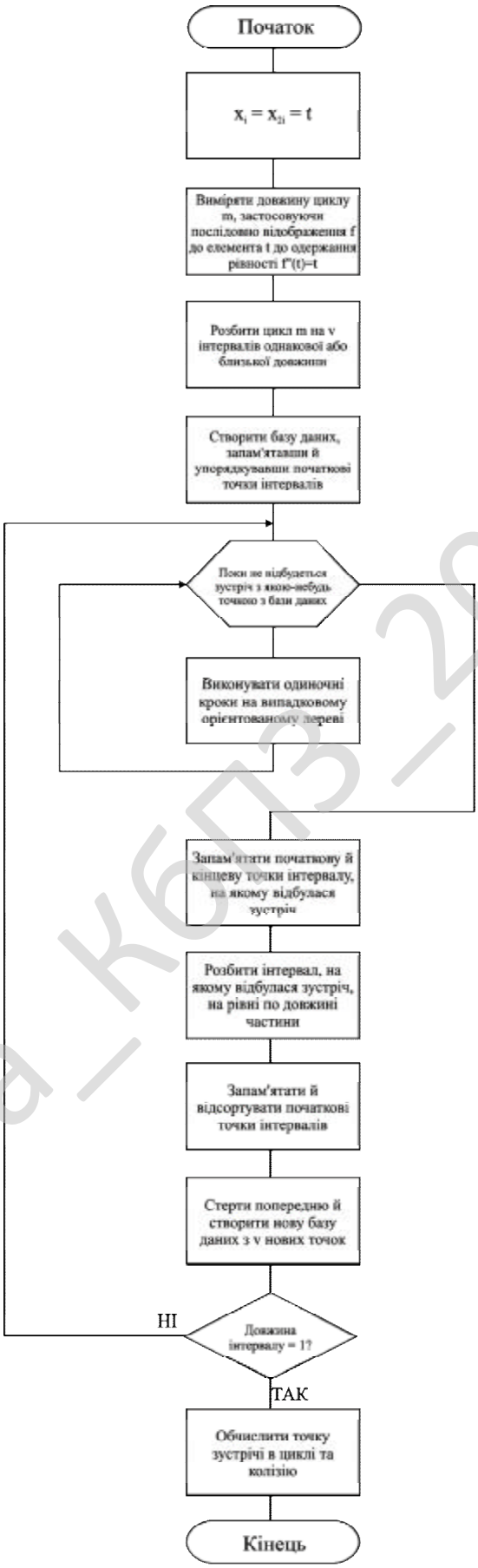

Рисунок 4.4 – Блок-схема алгоритму роботи підпрограми криптоаналізу

6. Повторити процедури пп.4,5 доти, поки не вийде довжина інтервалу, рівна 1. Обчислити точку зустрічі в циклі, обчислити колізію як пари вершин, одна з яких лежить у циклі, а друга – ні.

Цей алгоритм вимагає багаторазового виконання $O(\sqrt{\#M})$ кроків до входу в цикл і виконання сортування бази даних. На кожному етапі при створенні нової бази даних довжина інтервалу скорочується в v раз, тобто кількість повторів дорівнює $O(\log_v \#M)$. Якщо $v << \sqrt{\#M}$, то складністю сортування бази даних можна знехтувати. Тоді складність алгоритму дорівнює $O(\sqrt{\#M}\log_v \#M)$.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті. У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Дана система призначена для системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI. Програмне забезпечення розроблялося для персональної обчислювальної техніки не нижче Intel Pentium III 500 Mhz з наступними характеристиками:

- обсяг ОЗП не нижче 128 Mb;
- графічний адаптер SVGA;
- маніпулятор типу "миша";
- WINDOWS 10/11.

Програма має зручний та інтуїтивно зрозумілий інтерфейс. Головне вікно програми зображене на рисунку 5.1.

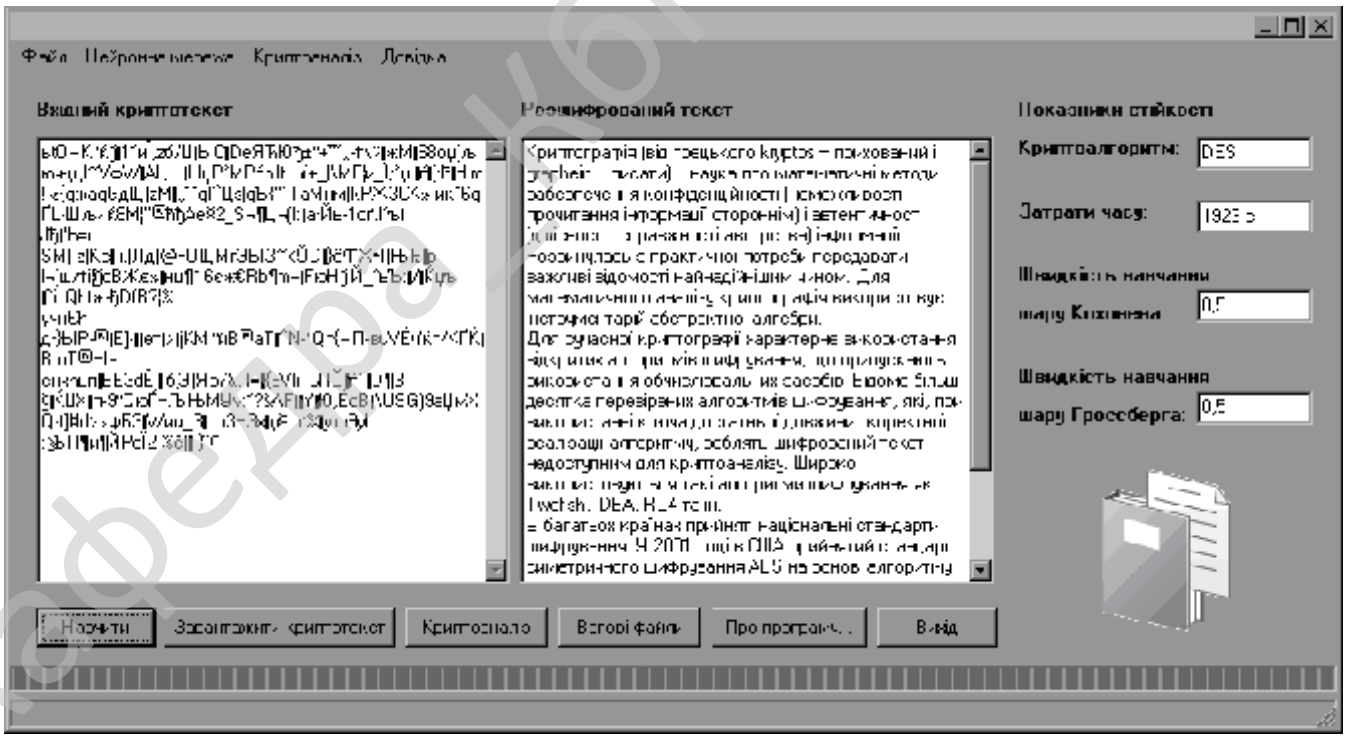

Рисунок 5.1 – Головне вікно програми

Для початку роботи із програмою необхідно запустити crypt.exe.

Щоб завантажити навчальну вибірку потрібно натиснути на кнопку «Навчити» і в запропонованому діалозі вибрати файл із розширенням *.edu. Щоб вибрати криптотекст потрібно натиснути кнопку «Завантажити криптотекст» і в запропонованому діалозі вибрати необхідний файл. Після цього його зміст з'явиться у полі «Вхідний криптотекст». Щоб завантажити ваговий файл потрібно натиснути кнопку «Вагові файли» і в запропонованому діалозі вибрати файл із розширенням *.wes. При відкритті файлів з неправильною внутрішньою структурою програма видає повідомлення користувачеві: «Неправильна структура файлу перевірте файл або виберіть інший».

Після натискання кнопки «Криптоаналіз» у полі «Розшифрований текст» з'являється результат роботи програми.

Щоб завершити роботу із програмою необхідно натиснути на кнопку «Вихід» або на «хрестик» у правому верхньому куті робочого вікна програми.

У правому вікні програми розташовані показники стійкості криптографічного алгоритму. До них відносяться затрати часу на криптоаналіз, швидкість навчання шару Кохонена та швидкість навчання шару Гроссберга. Також програма виводить назву використаного для шифрування файлу криптоалгоритму.

Переглянути короткі відомості про програму можна натиснувши кнопку «Про програму…» (рисунок 5.2).

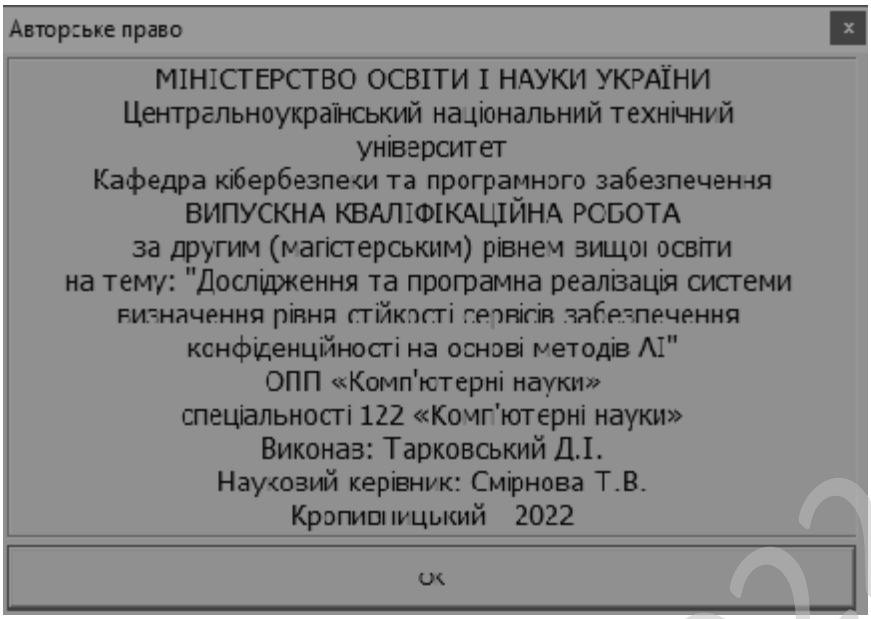

Рисунок 5.2 – Вікно довідки

На рисунках 5.3 – 5.6 наведено скриншоти усіх режимів роботи програми, що обираються за допомогою меню користувача.

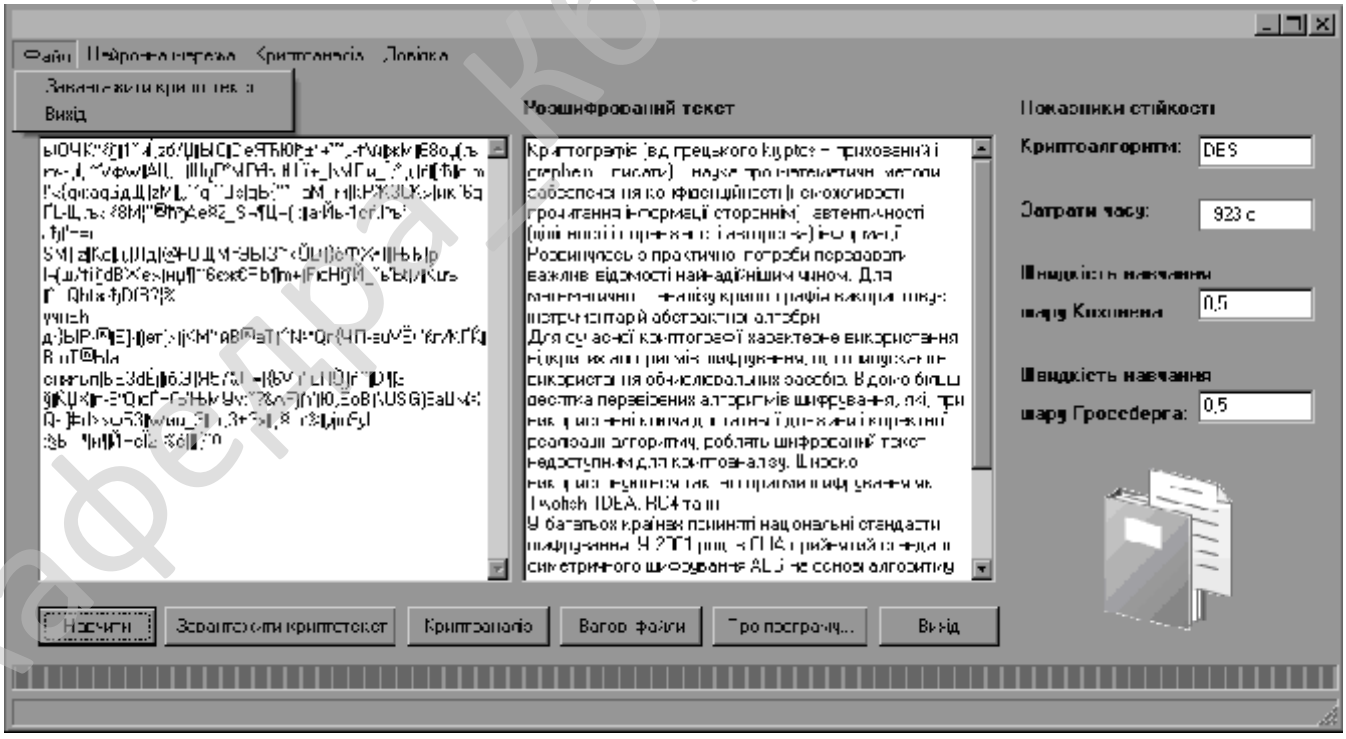

Рисунок 5.3 – Пункти меню **Файл**

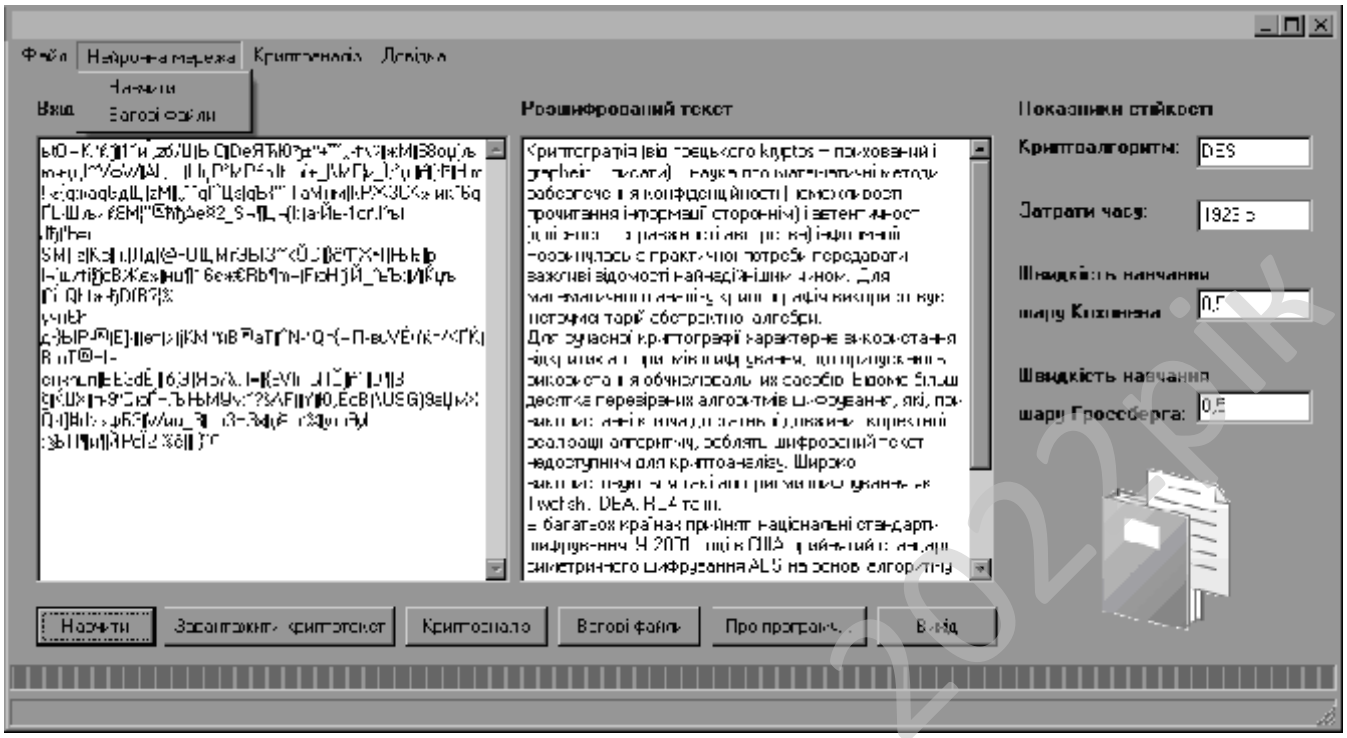

Рисунок 5.4 – Пункти меню **Нейронна мережа**

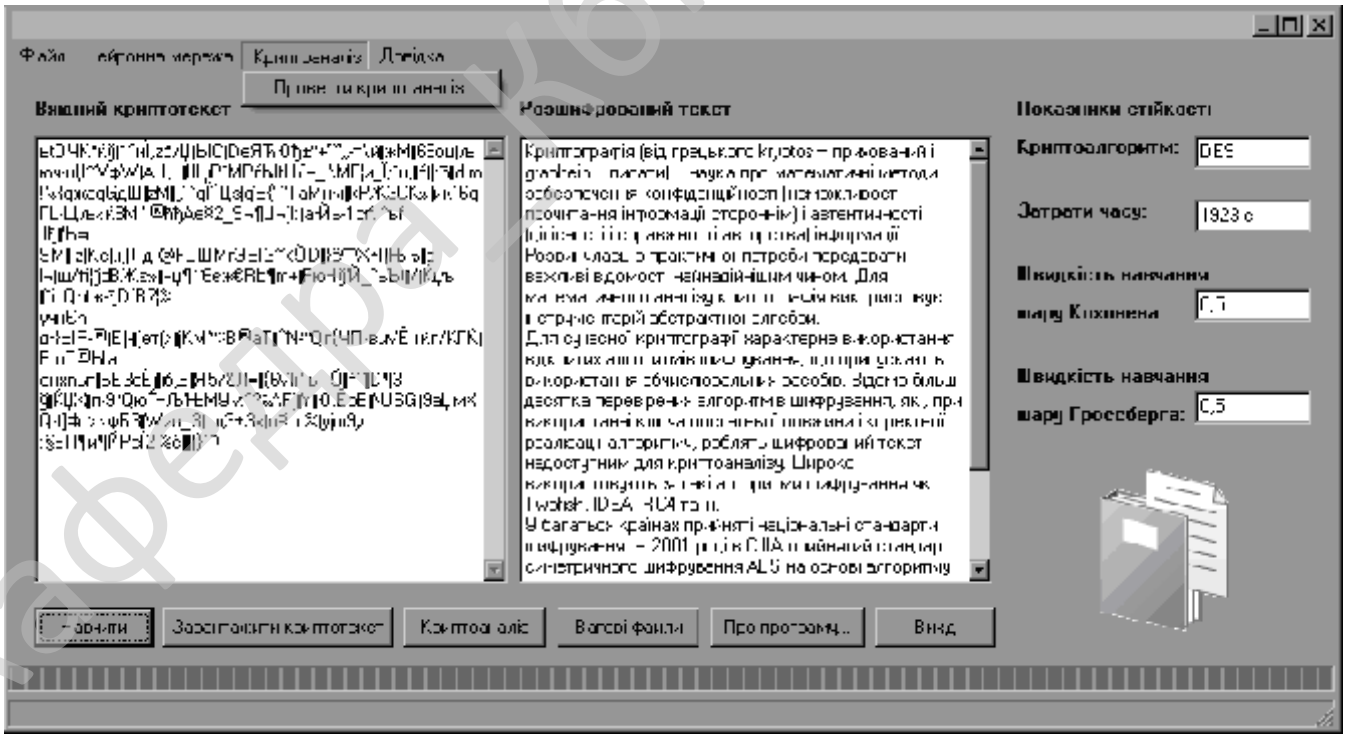

Рисунок 5.5 – Пункти меню **Криптоаналіз**

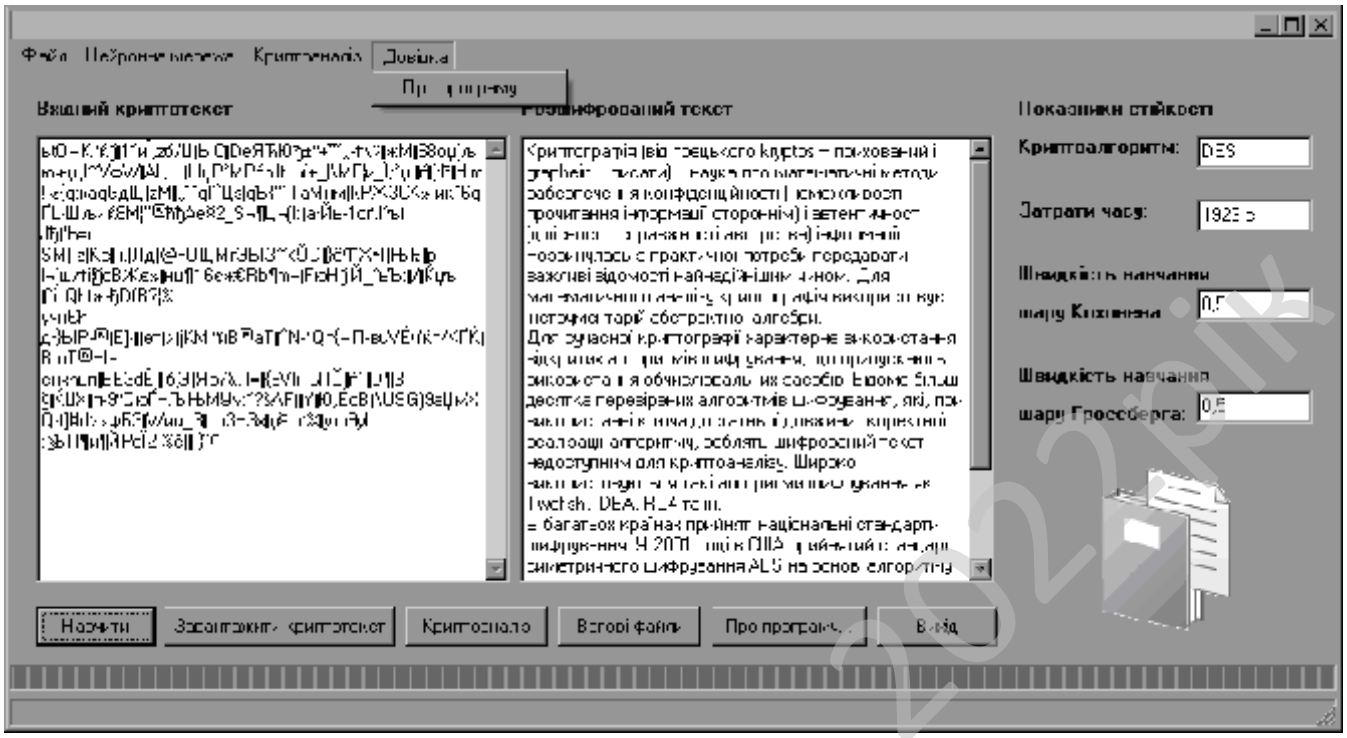

Рисунок 5.6 – Пункти меню **Довідка**

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Метою розробки є дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Об'єктом дослідження є процес визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Предметом дослідження є методи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Методи дослідження базуються на методах захисту інформації та штучного інтелекту (AI), методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– Розроблено вітчизняний продукт визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI, 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 визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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)%.

$$T_{РП} = 0{,}3 \cdot 3{,}23 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 77 = 157 \text{ люд/день.}$$

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	157	Ф 7.1-7.4
Впровадження	13	Д13
Всього	198	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{198 \cdot 1}{60 - 5} = 3,6 \text{ ставки.}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	100	250	4,17
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	35,33

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{35 \cdot 3}{1,2} = 87,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел} = 87,5/(60 \cdot 8) = 0,18$ 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)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,6	8084,5	87312,6
Інженер-електронщик	0,18	7000	3780
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,5	7000	10500
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	9000	13500
Всього за період розробки	$R_{сп}$ = 7,03	-	$Ф_{роб}$ = 178842,6

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.

$$З_{сд} = \frac{178843}{7,03 \cdot 60} = 424 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

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

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

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

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

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

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

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Компбест за 06.11.22 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i7-4770K 3.5GHz/5GT/s/8MI s1150	-
Системна плата	MSI H81M-S03 (s1150, DDR3, USB 3.(INTEL H81 , PCI-Ex16) з підтримкоі відеоядра процесора	-
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	-
Жорсткий диск	240 GB SSD	-
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC(Registered (M393B1K70CH0-CH9Q5)	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	-
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	-

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.

$$З_о = 424 \cdot 198/280 = 300 \text{ грн.}$$

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:

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

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

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 – 22,4 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.):

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

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:

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

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

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%.

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

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 7.9.

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	$З_р$	25190	20032
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	26678	21369

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 25190 грн до 20032 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.

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}5 \text{ роки}$$

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.

$$E_{сп} = (26678\text{-}21369)\text{-}0{,}25 \cdot 1226 = 5003 \text{ грн.}$$

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	4,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	5003
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,23

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

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

$$T_{сп} = \frac{1226}{26678 - 21369} = 0,23 \ року .$$

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

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ОЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5
Довжина	5,95
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7,4
Об'єм, V	$м^3$	не менше 20.0	20,8

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і

норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таким чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого

типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

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;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

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).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5 м, довжина – 5,95 м, висота – 2,8 м.

У зазначеному приміщенні працює 4 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою [1]:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

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

S – площа освітлюваного приміщення (у нашому випадку S=5 х 5,95 = 29,7 м^2);

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

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

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

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

де:

S – площа приміщення, *S* = 29,7 м2;

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

A – ширина приміщення, *A* = 5 м;

B – довжина приміщення, *B* = 5,95 м.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$i=1,4.$$

Знаючи індекс приміщення, за знаходимо *n* = 0,29 (з табличних даних коефіцієнтів використання світлового потоку (*n*) світильників з відповідним типом лампам) [8]. Підставимо всі значення у формулу, визначемо світловий потік: *F=64027 Лм.*

Для розрахунку дудемо використовувати світлодіодні стельові панелі *Призма-72 6400К*, світловий потік яких $F_л$ = *7200 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де: *F* – світловий потік, $F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$N= 64027 / 7200=8,8 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 9 шт.

8.6 Висновки до розділу

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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 визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– Досліджена система визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

– На основі отриманих результатів досліджень створена програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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

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

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

Програма реалізована на мові високого рівня Delphi 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

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 5003 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,23 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тарковський Д.І. Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.
2. Брюс Шнайдер Прикладная криптография 2-е изд. М: 2000 г.
3. Черемушкин А. В. Лекции по арифметическим алгоритмам в криптографии – М.: Издательство Московского центра непрерывного математического образования, 2000 г. С. 194-203.
4. Вороновский Г. К. Махотило А. В. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. – Харьков Основа 1997 г. С. 67-81
5. Torben Brauner Introduction to Linear Logic: BRICS, Department of Computer Science 1997
6. И. В. Заенцев Нейронные сети: основные модели Воронеж 1999 г. с. 12-64
7. 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).**
8. 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**).
9. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks.*

Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

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

11. 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).**

12. 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).**

13. 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).**

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

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

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

17. 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)**.

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

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

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

21. 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)**.

22. 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)**.

23. 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)**.

24. 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)**.

25. 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)**.

26. 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)**.

27. 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)**.

28. 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)**.

29. 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)**.

30. 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)**.

31. 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)**.

32. 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).

33. 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).

34. 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).

35. 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.

36. 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.

37. 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.

38. 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.

39. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в

інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022. (**Фахове видання. Категорія «Б»**)

40. 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.

41. 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.

42. 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

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

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

54. 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.

55. 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.

56. 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.

57. 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.

58. 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.

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

60. 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

61. 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

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

63. 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

64. 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

65. 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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

					ВКРМ-122.22.0014.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Тарковський Д.І.			Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів АІ	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнова Т.В.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КН-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI.

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 визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4.1.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 129 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнова Т.В.

Дослідження та програмна реалізація системи визначення рівня стійкості сервісів забезпечення конфіденційності на основі методів AI

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 89

Літера: РП

Кропивницький – 2022 року

Основна програма

Файл main.pas основної програми

```
unit Main;

interface

{$I VER.INC}

// Підключення бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls, Buttons, HCMngr, ComCtrls, DECUtil, RNG;

// Опис змінних

type
  TMainForm = class(TForm)
    MainMenu: TMainMenu;
    MItemFile: TMenuItem;
    MItemStats: TMenuItem;
    MItemHashMemory: TMenuItem;
    MItemHashFile: TMenuItem;
    MItemExit: TMenuItem;
    P1: TPanel;
    Bevel1: TBevel;
    LHash: TLabel;
    EHashFile: TEdit;
    LInputfile: TLabel;
    BtnHashFile: TBitBtn;
    CBHash: TComboBox;
    LHAlgorithm: TLabel;
    LHashInfo: TLabel;
    LBase16: TLabel;
    EBase16: TEdit;
    LBase64: TLabel;
    EBase64: TEdit;
    BtnCalcHash: TBitBtn;
    OpenDialog: TOpenDialog;
    LHashTimes: TLabel;
    LHashTime: TLabel;
    LCipher: TLabel;
    Bevel2: TBevel;
    LCInput: TLabel;
    LCAlgorithm: TLabel;
    LCipherInfo: TLabel;
    LCKey: TLabel;
    LHashKey: TLabel;
    LCTimes: TLabel;
    LEncodeTime: TLabel;
    ECipherFile: TEdit;
    BtnInputFile: TBitBtn;
    CBCipher: TComboBox;
    EKey: TEdit;
    EHashKey: TEdit;
    BtnCipher: TBitBtn;
    CBCipherMode: TComboBox;
    LCMode: TLabel;
    LModeInfo: TLabel;
    LCipherHint: TLabel;
    BtnViewHashFile: TBitBtn;
    BtnViewCipherFiles: TBitBtn;
    LDTimes: TLabel;
    LDecodeTime: TLabel;
```

```
    LHashInput: TLabel;
    EHashInput: TEdit;
    EHashDEC: TEdit;
    LHashDEC: TLabel;
    EHashENC: TEdit;
    LHashENC: TLabel;
    N2: TMenuItem;
    MItemTestFile: TMenuItem;
    MItemTestRes: TMenuItem;
    N1: TMenuItem;
    MItemCipherMemory: TMenuItem;
    MItemCipherFile: TMenuItem;
    MItemMemCBC: TMenuItem;
    MItemMemCTS: TMenuItem;
    MItemMemCFB: TMenuItem;
    MItemMemOFB: TMenuItem;
    MItemMemECB: TMenuItem;
    MItemFileCTS: TMenuItem;
    MItemFileCBC: TMenuItem;
    MItemFileCFB: TMenuItem;
    MItemFileOFB: TMenuItem;
    MItemFileECB: TMenuItem;
    MItemHashVector: TMenuItem;
    MItemCipherVector: TMenuItem;
    Progress: TProgressBar;
    MItemExamples: TMenuItem;
    MItemPart: TMenuItem;
    MItemStrings: TMenuItem;
    MItemIV: TMenuItem;
    CipherManager: TCipherManager;
    HashManager: THashManager;
    OneTimePassword1: TMenuItem;
    HowuseTProtectionClasses1: TMenuItem;
    procedure MItemExitClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure CBHashClick(Sender: TObject);
    procedure BtnCalcHashClick(Sender: TObject);
    procedure BtnHashFileClick(Sender: TObject);
    procedure CBCipherClick(Sender: TObject);
    procedure CBCipherModeClick(Sender: TObject);
    procedure BtnViewHashFileClick(Sender: TObject);
    procedure EHashFileChange(Sender: TObject);
    procedure BtnInputFileClick(Sender: TObject);
    procedure BtnViewCipherFilesClick(Sender: TObject);
    procedure ECipherFileChange(Sender: TObject);
    procedure BtnCipherClick(Sender: TObject);
    procedure EKeyChange(Sender: TObject);
    procedure MItemTestFileClick(Sender: TObject);
    procedure MItemTestResClick(Sender: TObject);
    procedure MItemHashSpeedClick(Sender: TObject);
    procedure MItemCipherMemSpeedClick(Sender: TObject);
    procedure MItemCipherFileSpeedClick(Sender: TObject);
    procedure MItemHashVectorClick(Sender: TObject);
    procedure MItemCipherVectorClick(Sender: TObject);
    procedure ManagerProgress(Sender: TObject; Current, Maximal: Integer);
    procedure FormActivate(Sender: TObject);
    procedure MItemPartClick(Sender: TObject);
    procedure MItemStringsClick(Sender: TObject);
    procedure MItemIVClick(Sender: TObject);
    procedure OneTimePassword1Click(Sender: TObject);
    procedure HowuseTProtectionClasses1Click(Sender: TObject);
  private
    FTime: Comp;
    FViewer: String;
    FENCFile: String;
    FDECFile: String;
    FCreated: Boolean;
    function SelectFile(Edit: TEdit): Boolean;
    procedure ExecuteView(const FileName: String);
```

```
    function Counter(Size: Integer): String;
  public
  end;

var
  MainForm: TMainForm;

implementation

// Початок опису реалізацій функцій нейронного аналізу

uses ShellAPI, Hash, Cipher, ResFrm, MemSpd, ClipBrd, PCrypt, Strdemo,
     IVDemo, RFC2289, OTPDemo, GenForm;

{$R *.DFM}

// визначення розміру файлу, який дається на аналіз

function GetFileSize(const Filename: String): Integer;
var
  SR: TSearchRec;
begin
  if FindFirst(Filename, faAnyFile, SR) = 0 then Result := SR.Size
    else Result := -1;
  FindClose(SR);
end;

procedure TMainForm.ExecuteView(const FileName: String);
begin
  if FileExists(FileName) then
    ShellExecute(Handle, nil, PChar(FViewer), PChar(Filename), nil,
sw_ShowNormal);
end;

// Вибір файлу

function TMainForm.SelectFile(Edit: TEdit): Boolean;
begin
  OpenDialog.InitialDir := ExtractFilePath(Edit.Text);
  if OpenDialog.InitialDir = '' then OpenDialog.InitialDir :=
ExtractFilepath(ParamStr(0));
  Result := OpenDialog.Execute;
  if Result then Edit.Text := OpenDialog.FileName;
end;

function TMainForm.Counter(Size: Integer): String;
var
  S: Double;
begin
  if Size >= 0 then
  begin
    FTime := PerfCounter - FTime;
    S := FTime / PerfFreq;
    Result := FormatFloat('#,###0.0000 Seconds,  ', S) +
              FormatFloat('#0 ms,   ', S * 1000) +
              Format('%d Bytes, %s Kb Datasize  ', [Size,
FormatFloat('#,##0.00', Size / 1024)]) +
              FormatFloat('ca #,##0.00 Mb/sec', (1 / S) * (Size / (1024 *
1024)));
  end else
  begin
    Result := '';
    FTime := PerfCounter;
  end;
end;

// обробка нажаття клавіші

procedure TMainForm.MItemExitClick(Sender: TObject);
```

```
begin
  Close;
end;

//Створення головної форми

procedure TMainForm.FormCreate(Sender: TObject);
begin
  if not DECUtil.InitTestIsOk then Caption := 'Init Test failed';
  FViewer := 'notepad.exe';
  FENCFile := ChangeFileExt(ParamStr(0), '.enc');
  FDECFile := ChangeFileExt(ParamStr(0), '.dec');
  EHashFile.Text := ParamStr(0);
  ECipherFile.Text := EHashFile.Text;

  HashNames(CBHash.Items);
  CipherNames(CBCipher.Items);
end;

procedure TMainForm.FormActivate(Sender: TObject);
begin

  if not FCreated then
  begin
    Update;
    CBHash.ItemIndex := 0;
    CBCipherMode.ItemIndex := 0;
    CBCipherModeClick(CBCipherMode);
    CBCipher.ItemIndex := 0;
    FCreated := True;
    CBHashClick(CBHash);
    CBCipherClick(CBCipher);
  end;
end;

procedure TMainForm.CBHashClick(Sender: TObject);
begin
  CBHash.ItemIndex := CBHash.ItemIndex;

{1. Варіант вибору алгоритму хеш функції для аналізу}
  HashManager.Algorithm := CBHash.Text;
  LHashInfo.Caption := HashManager.Description + ', ' +
                       HashManager.HashClass.ClassName;

{2.Варіант }
//  HashManager.HashClass := THashClass(CBHash.Items.Objects[CBHash.ItemIndex]);
//  LHashInfo.Caption := Format('%s, %d bit Digestsize',
//    [HashManager.HashClass.ClassName, HashManager.HashClass.DigestKeySize *
8]);

// Тестування хеш-функції на корректість
  try
    if not HashManager.HashClass.SelfTest then
      MessageBox(Handle, 'Self Test failed', 'Hash Self Test', mb_Ok);
  except
    Application.HandleException(Self);
  end;
  BtnCalcHashClick(nil);
end;

procedure TMainForm.BtnCalcHashClick(Sender: TObject);
var
  FileSize: Integer;
//  Hash: THash;
//  HashClass: THashClass;
//  Digest: String;
//  Stream: TFileStream;
//  Buf: array[0..7] of Integer;
//  Len: Integer;
```

```
begin
  EKeyChange(nil);
  EBase16.Text := '';
  EBase64.Text := '';
  FileSize := GetFileSize(EHashFile.Text);
  if (FileSize >= 0) and FCreated then
  try
      Screen.Cursor := crHourglass;
      Application.ProcessMessages;
      Counter(-1);

    HashManager.CalcFile(EHashFile.Text);

      LHashTime.Caption := Counter(FileSize);
      EBase16.Text := HashManager.DigestString[fmtHEX];
      EBase64.Text := HashManager.DigestString[fmtMIME64];

  finally
    Screen.Cursor := crDefault;
  end;
end;

procedure TMainForm.BtnHashFileClick(Sender: TObject);
begin
  if SelectFile(EHashFile) then BtnCalcHashClick(nil);
end;

procedure TMainForm.BtnViewHashFileClick(Sender: TObject);
begin
  ExecuteView(EHashFile.Text);
end;

procedure TMainForm.EHashFileChange(Sender: TObject);
begin
  BtnViewHashFile.Enabled := FileExists(EHashFile.Text);
  BtnCalcHash.Enabled := FileExists(EHashFile.Text);
end;

procedure TMainForm.CBCipherClick(Sender: TObject);
begin
{скоректуємо вибір Display з Combobox де вибір з VK_UP або VK_DOWN}
  CBCipher.ItemIndex := CBCipher.ItemIndex;


{1. Варіант визначення шифру}
  CipherManager.Algorithm := CBCipher.Text;

  LCipherInfo.Caption := CipherManager.Description + ', ' +
                         CipherManager.CipherClass.ClassName;

{2.Variant }
//  CipherManager.CipherClass :=
TCipherClass(CBCipher.Items.Objects[CBCipher.ItemIndex]);

//  LCipherInfo.Caption := Format('%s, %d bit MaxKeysize',
//     [CipherManager.CipherClass.ClassName, CipherManager.CipherClass.KeySize *
8]);


// Тестування шифру на коректність результату
  try
    if not CipherManager.CipherClass.SelfTest then
      MessageBox(Handle, 'Self Test failed', 'Cipher Self Test', mb_Ok);
  except
// Abstract Error when TCipher.TestVector not анульовано
    Application.HandleException(Self);
  end;
  BtnCipherClick(nil);
end;
```

```
procedure TMainForm.CBCipherModeClick(Sender: TObject);
const
  sMode : array[TCipherMode] of String =
    ('Cipher Text Stealing', 'Cipher Block Chaining', 'Cipher Feedback',
     'Output Feedback', 'Electronic Code Book', 'CBC MAC', 'CTS MAC', 'CFB
MAC');
begin
  CipherManager.Mode := TCipherMode(CBCipherMode.ItemIndex);
  LModeInfo.Caption := sMode[CipherManager.Mode];
  BtnCipherClick(nil);
end;

procedure TMainForm.BtnInputFileClick(Sender: TObject);
begin
  if SelectFile(ECipherFile) then BtnCipherClick(nil);
end;

procedure TMainForm.BtnViewCipherFilesClick(Sender: TObject);
begin
  ExecuteView(ECipherFile.Text);
  ExecuteView(FENCFile);
  ExecuteView(FDECFile);
end;

procedure TMainForm.ECipherFileChange(Sender: TObject);
begin
  BtnViewCipherFiles.Enabled := FileExists(ECipherFile.Text);
  BtnCipher.Enabled := FileExists(ECipherFile.Text);
end;

procedure TMainForm.EKeyChange(Sender: TObject);
begin
{Автоматичне коректування Display, значення хеш  Key}
{Використовуємо вибраний HashClass, це тотожне для вибору CipherManager для
кодування / декодування}
  EHashKey.Text := HashManager.HashClass.CalcString(EKey.Text, nil, fmtHEX);

{ Це показує закодований Hashvalue}
{
  CipherManager.InitKey(EKey.Text, nil);
  EHashKey.Text := CipherManager.Cipher.Hash.DigestBase16;
}
end;

procedure TMainForm.BtnCipherClick(Sender: TObject);
var
  FileSize: Integer;
begin
  EHashInput.Text := '';
  EHashENC.Text   := '';
  EHashDEC.Text   := '';
  FileSize := GetFileSize(ECipherFile.Text);
  if (FileSize > 0) and FCreated then
  try
      Screen.Cursor := crHourGlass;
      Application.ProcessMessages;

// ініціалізуємо ключ
    CipherManager.InitKey(EKey.Text, nil);
      Counter(-1);
// Декодуємо вхідний файл до файлу навчання Demo.enc
    CipherManager.EncodeFile(ECipherFile.Text, FENCFile);
      LEncodeTime.Caption := Counter(FileSize);

// ініціалізуємо ключ
    CipherManager.InitKey(EKey.Text, nil);
//  CipherManager.InitKey(EKey.Text + 'Bad Key', nil);
```

```
//  Замість CipherManager.InitKey потрібно
//  CipherManager.Cipher.Done;
      Counter(-1);
// Декодуємо Demo.enc до Demo.dec
    CipherManager.DecodeFile(FENCFile, FDECFile);

      LDecodeTime.Caption := Counter(FileSize);

// Перевіряємо який  процесс хешування використовується
      EHashInput.Text := THash_MD4.CalcFile(ECipherFile.Text, nil, fmtDEFAULT);
      EHashENC.Text   := THash_MD4.CalcFile(FENCFile, nil, fmtDEFAULT);
      EHashDEC.Text   := THash_MD4.CalcFile(FDECFile, nil, fmtDEFAULT);
      if EHashInput.Text <> EHashDEC.Text then EHashDEC.Color := clRed
        else EHashDEC.Color := clBtnHighlight;
  finally
    Screen.Cursor := crDefault;
  end;
end;

// Підпрограмма тестування файла навчання

procedure TMainForm.MItemTestFileClick(Sender: TObject);
const
  BufSize = 1024 * 4;
var
  P: PByteArray;
  Start,Stop: Comp;
begin
  EHashFile.Text := ChangeFileExt(ParamStr(0), '.tst');
  ECipherFile.Text := EHashFile.Text;
  GetMem(P, BufSize);
  try
    Screen.Cursor := crHourGlass;
    with TFileStream.Create(EHashFile.Text, fmCreate) do
    try
      RND.Protection := TCipher_SCOP.Create('Password', nil);
      RND.Seed('', -1);  // Повністю випадковий
      Start := PerfCounter;
      repeat
        RND.Buffer(P^, BufSize); // Запвнюємо буфер випадковими даними
        Write(P^, BufSize);
      until Position >= 1024 * 1024;
      Stop := PerfCounter;
    finally
      Free;
      RND.Protection := nil; // Звільняємо від захисту
    end;
  finally
    Screen.Cursor := crDefault;
    FreeMem(P, BufSize);
  end;
  Start := Stop - Start;
  Stop := PerfFreq;
  MessageDlg('1Mb in ' + FloatToStr(Start / Stop) + ' Secs filled with secure
random Data.',
              mtInformation, [mbOk], 0);
  EHashFileChange(EHashFile);
  ECipherFileChange(ECipherFile);
  BtnCalcHashClick(nil);
  BtnCipherClick(nil);
end;

procedure TMainForm.MItemTestResClick(Sender: TObject);
begin
  with TCheckResForm.Create(Self) do
  try
    ShowModal;
  finally
    Free;
```

```
  end;
end;

// Підпрограма обробки швидкості хешування

procedure TMainForm.MItemHashSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(Sender = MItemHashMemory, True, cmECB);
end;

procedure TMainForm.MItemCipherMemSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(True, False, TCipherMode(TComponent(Sender).Tag));
end;

// Підпрограма обробки швидкості шифрування

procedure TMainForm.MItemCipherFileSpeedClick(Sender: TObject);
begin
  with TSpeedForm.Create(Self) do
    Execute(False, False, TCipherMode(TComponent(Sender).Tag));
end;

// Визначаємо фрагмент якого коду обробляється

procedure MakeCodeFragment(const Data: String; Len: Integer);
var
  C: String;
  I: Integer;
begin
  C := '         MOV   EAX,OFFSET @Vector' + #13#10 +
       '         RET'                      + #13#10 +
       '@Vector: ';
  for I := 0 to Len -1 do
  begin
    if I mod 8 = 0 then
    begin
      if I > 0 then C := C + #13#10 + '         ';
      C := C + 'DB    ';
    end else C := C + ',';
    C := C + IntToHex(Byte(Data[I+1]), 3) + 'h';
  end;
  Clipboard.AsText := C;
end;

procedure TMainForm.MItemHashVectorClick(Sender: TObject);
{генеруємоTestVector для хеш та вставляємо Codefragment to Clipboard}
var
  Data,Caption: String;
begin
  with HashManager.HashClass do
  begin
    Data := CalcBuffer(GetTestVector^, 32, nil, fmtCOPY);
    MakeCodeFragment(Data, DigestKeySize);
    Caption := 'Testvector for ' + ClassName;
    Data    := StrToFormat(PChar(Data), DigestKeySize, fmtHEX);
    MessageBox(Handle, PChar(Data), PChar(Caption), mb_Ok);
  end;
end;

procedure TMainForm.MItemCipherVectorClick(Sender: TObject);
{ генеруємо TestVector для шифру та вставляємо Codefragment to Clipboard }
var
  Data, Caption: String;
begin
  with CipherManager.CipherClass.Create('', nil) do
  try
```

```
    Data := ClassName;
    Mode := cmCTS;
    Init(PChar(Data)^, Length(Data), nil);
    SetLength(Data, 32);
    EncodeBuffer(GetTestVector^, PChar(Data)^, 32);
    MakeCodeFragment(Data, 32);
    Caption := 'Testvector for ' + ClassName;
    Data    := StrToFormat(PChar(Data), 32, fmtHEX);
    MessageBox(Handle, PChar(Data), PChar(Caption), mb_Ok);
  finally
    Free;
  end;
end;

procedure TMainForm.ManagerProgress(Sender: TObject; Current, Maximal: Integer);
begin
{Визначаємо шифр або хеш
  TCipher_xxx.En/DecodeFile(), TCipher_xxx.En/DecodeStream()
  THash_xxx.CalcStream(), THash_xxx.CalcFile()}
{$IFDEF VER_D3H}
  Progress.Max := Maximal;
  Progress.Position := Current;
{$ELSE}
  if Maximal <= 0 then Progress.Position := 0
    else Progress.Position := Trunc(Progress.Max / Maximal * Current)
{$ENDIF}
{finished is by Current = 0 and Maximal = 0}
end;

// Процедури обробки натискання клавіш

procedure TMainForm.MItemPartClick(Sender: TObject);
begin
  with TPartForm.Create(Self) do Show;
end;

procedure TMainForm.MItemStringsClick(Sender: TObject);
begin
  with TStringForm.Create(Self) do Show;
end;

procedure TMainForm.MItemIVClick(Sender: TObject);
begin
  with TIVForm.Create(Self) do Show;
end;

procedure TMainForm.OneTimePassword1Click(Sender: TObject);
begin
  with TOTPForm.Create(Self) do Show;
end;
procedure TMainForm.HowuseTProtectionClasses1Click(Sender: TObject);
begin
  with TGForm.Create(Self) do Show;
end;
end.
```

GenForm.pas – Побудова форм та основних обробників клавіш

```
unit GenForm;

interface

// Підключення бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Menus, ComCtrls, DECUtil, Hash, Cipher, RNG, RFC2289, ShellAPI,
  Sample, Cipher1;

// Опис головного об'єкту

type
  TGForm = class(TForm)
    MainMenu: TMainMenu;
    HashMAC: TMenuItem;
    M: TRichEdit;
    THashXXX: TMenuItem;
    MACwithRFC1: TMenuItem;
    ViewRFC2202html1: TMenuItem;
    N1: TMenuItem;
    N2: TMenuItem;
    UsingfromHashs1: TMenuItem;
    File1: TMenuItem;
    Exit1: TMenuItem;
    N3: TMenuItem;
    MItemFormats: TMenuItem;
    TCipherXXX: TMenuItem;
    TRandomXXX: TMenuItem;
    UsingfromCiphers1: TMenuItem;
    N4: TMenuItem;
    CipherMAC: TMenuItem;
    TransactionNumbersTANs1: TMenuItem;
    UsingfromRandoms1: TMenuItem;
    procedure HashMACClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure MACwithRFC2104Click(Sender: TObject);
    procedure ViewRFC2202html1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure UsingfromHashs1Click(Sender: TObject);
    procedure UsingfromCiphers1Click(Sender: TObject);
    procedure TransactionNumbersTANs1Click(Sender: TObject);
    procedure CipherMACClick(Sender: TObject);
    procedure UsingfromRandoms1Click(Sender: TObject);
  private
    Format: Integer; // the used String Format
    procedure DoInfo(const Value: String; Color: TColor);
    procedure FormatClick(Sender: TObject);
  public
  end;

var
  GForm: TGForm;

implementation

{$R *.DFM}
const
  sSelfTest : array[Boolean] of String = ('failed', 'success');

//Початок роботи підпрограми

procedure TGForm.DoInfo(const Value: String; Color: TColor);
begin // Показуємо Value в Color в Richedit
  M.SelStart := MaxInt div 16;
```

```
  M.SelLength := 0;
  M.SelAttributes.Color := Color;
  M.Lines.Add(Value);
  M.SelAttributes.Color := clWindowText;
  M.Perform(em_ScrollCaret, 0, 0);
  M.Update;
end;

// Обробник закриття форм

procedure TGForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TGForm.FormatClick(Sender: TObject);
begin
  with TMenuItem(Sender) do
  begin
    Checked := True;
    Format := Tag;
    DoInfo('Строка Displayformat змінена на: ' + Caption, clRed);
  end;
end;

procedure TGForm.FormCreate(Sender: TObject);
var
  I: Integer;
  S: TStringList;
  FMT: TStringFormatClass;
  MI: TMenuItem;
begin
// Встановлюємо внутрішній стан
  Format := fmtHEXVIEW;

  M.HandleNeeded;
  M.Paragraph.Tab[0] := 90;

  S := TStringList.Create;
  try
    GetStringFormats(S);
    for I := 0 to S.Count-1 do
    begin
      FMT := TStringFormatClass(S.Objects[I]);
      if (FMT.Format = fmtCOPY) or
         (FMT.Format = fmtSAMPLE) then Continue;
      MI := TMenuItem.Create(MItemFormats);
      MI.Caption := FMT.Name;
      MI.Tag := FMT.Format;
      MI.OnClick := FormatClick;
      MI.RadioItem := True;
      MI.Checked := FMT.Format = Format;
      MItemFormats.Add(MI);
    end;
  finally
    S.Free;
  end;
end;

procedure TGForm.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TGForm.UsingfromHashs1Click(Sender: TObject);
var
  HUser: THashClass;
  S: String;
  Buffer: array[0..127] of Byte;
```

```
  I: Integer;
  Stream: TStream;
  Cipher: TCipher;
begin
  M.Clear;
  HUser := THash_RipeMD128; // змінюємо для інших хеш-функцій

  for I := Low(Buffer) to High(Buffer) do Buffer[I] := I; // встановлюємо Buffer

  DoInfo('Використовується хеш', clRed);
  DoInfo('Користувач визнчів хеш: ' + GetHashName(HUser), clMaroon);
  DoInfo('Початок криптоаналізу хешу користувача', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('1. Індивідуальні дані з ParamStr(0), DEMO.EXE', clBlue);


  S := THash_MD5.CalcFile(ParamStr(0), nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcFile(ParamStr(0), nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('2. Індивідуальні дані з строки, "Test"', clBlue);

  S := THash_MD5.CalcString('Test', nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcString('Test', nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('3. Індивідуальні дані з буферу', clBlue);

  S := THash_MD5.CalcBuffer(Buffer, SizeOf(Buffer), nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcBuffer(Buffer, SizeOf(Buffer), nil, Format);
  DoInfo('HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('4. Індивідуальні дані з TStream, 1024 Bytes from DEMO.EXE at Position
123', clBlue);

  Stream := TFileStream.Create(ParamStr(0), fmOpenRead or fmShareDenyNone);
  try
    Stream.Position := 123;
    S := THash_MD5.CalcStream(Stream, 1024, nil, Format);
    DoInfo('MD5'#9+S, clWindowText);

    Stream.Position := 123;
    S := HUser.CalcStream(Stream, 1024, nil, Format);
    DoInfo('HUser'#9+S, clWindowText);

  finally
    Stream.Free;
  end;

//-----------------------------------------------------------------------------
  DoInfo('5. Використовувати любу хеш-функцію', clBlue);

  with THash_MD5.Create(nil) do
  try
    Init;
// встановлюємо Initial Digest, XOR's a Passphrase з Digest, повинно бути після
виклику Init
```

```
    S := 'Password';
    for I := 0 to Length(S)-1 do
      PByteArray(DigestKey)[I mod DigestKeySize] := PByteArray(DigestKey)[I mod
DigestKeySize] xor Byte(S[I+1]);

    Calc(Buffer[ 0], 16);   // розраховуємо перші 16 байтів з Buffer
    Calc(Buffer[33], 15);   // розраховуємо з Buffer[33] 15 байт
    for I := 1 to 11 do     // розраховуємо 11 станів для "Проміжний пароль"
      Calc('DEC Part I', 10);
    Calc(Buffer[99],  20);
    Done;

    S := DigestStr(Format);
    DoInfo('MD5'#9+S, clWindowText);

  finally
    Free;
  end;
// для добавлення хеш функцій на криптаналіз
  with HUser.Create(nil) do
  try
    Init;
// встановлюємо Initial Digest, XOR's a Passphrase з Digest, повинно бути після
Init
    S := 'Password';
    for I := 0 to Length(S)-1 do
      PByteArray(DigestKey)[I mod DigestKeySize] := PByteArray(DigestKey)[I mod
DigestKeySize] xor Byte(S[I+1]);

    Calc(Buffer[ 0], 16);   // calc the first 16 Bytes from Buffer
    Calc(Buffer[33], 15);   // calc from Buffer[33] 15 Bytes
    for I := 1 to 11 do
      Calc('DEC Part I', 10);
    Calc(Buffer[99],  3);
    Done;

    S := DigestStr(Format);
    DoInfo('HUser'#9+S, clWindowText);

  finally
    Free;
  end;
//---------------------------------------------------------------------------
  DoInfo('6. використовуємо TProtection метод для хеш', clBlue);

  with THash_MD5.Create(nil) do
  try
//---------------------------------------------------------------------------
// використовуємо MD5 кодування декодування:
// розраховуємо перші, Initialseed S0 (Password), рахуємо S0->S1->S2 та так
далі,//
    DoInfo('MD5 зашифровано, PlainText: "ваш текст наступний", Password "DEC"',
clGreen);
    Protection := TMAC_RFC2104.Create('DEC', nil); // Your Password

    S := CodeString('Ваш текст наступний', paEncode, Format);
    DoInfo('encoded'#9+S, clWindowText);

    S := CodeString(S, paDecode, Format);
    DoInfo('decoded'#9+S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('MD5 зашифрований, PlainText: "Ваш текст наступний", Password "DED"',
clGreen);
    Protection := TMAC_RFC2104.Create('DED', nil); // ваш пароль
//    Protection := TCipher_Blowfish.Create('DED', nil);

    S := CodeString('Ваш текст наступний', paEncode, Format);
    DoInfo('1. encoded'#9+S, clWindowText);
```

```
    S := CodeString(S, paDecode, Format);
    DoInfo('1. decoded'#9+S, clWindowText);

// обратно зашифрований paEncode/paDecode обміном,
// при заміні захисту на Blowfish ви побачите цей результат
    S := CodeString('Ваш текст наступний', paDecode, fmtCOPY); // Format must be
fmtCOPY
    DoInfo('2. encoded'#9+StrToFormat(PChar(S), Length(S), Format),
clWindowText);

    S := CodeString(S, paEncode, fmtCopy);
    DoInfo('2. decoded'#9+S, clWindowText);

//---------------------------------------------------------------------------
// paScramble, it's a One Way Function
    DoInfo('MD5 Scramble, Data: "Проміжні дані"', clGreen);
    Protection := TMAC.Create('DEC Scramble', nil); // ваш пароль

    S := CodeString('Проміжні дані', paScramble, Format);
    DoInfo('1. scramble'#9+S, clWindowText);
    S := CodeString('Проміжні дані', paScramble, Format);
    DoInfo('2. scramble'#9+S, clWindowText);

//---------------------------------------------------------------------------
// paWipe, - один з видів Function (paScramble) для видачі усіх інших
результатів
// Захист не потрібен при використанні корректних CodeBuffer, CodeFile,
CodeStream
// Для того, щоб убрати слабкі параметри використовується CodeString()

    DoInfo('MD5 Взломано, Data: "Дані взломані"', clGreen);
    Protection := nil;

    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('1. wiped'#9+S, clWindowText);
    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('2. wiped'#9+S, clWindowText);
    S := CodeString('Дані взломані', paWipe, Format);
    DoInfo('3. wiped'#9+S, clWindowText);

//---------------------------------------------------------------------------
// calculate a MD5 Fingerprint over Blowfish encrypted DEMO.EXE
// THash_MD5.CalcFile() с  Blowfish шифром розраховується
// MD5 автентифікатор.
// Цей метод взламує DEMO.EXE, розраховуючи MD5 та взламує MD5 Final Digest
    DoInfo('MD5 розраховується над Blowfish взламуючи DEMO.EXE', clGreen);

    Protection := TCipher_Blowfish.Create('DEC', nil);
    CodeFile(ParamStr(0), '', paCalc);

    DoInfo('MD5-Digest'#9+DigestStr(Format), clWindowText);

//---------------------------------------------------------------------------
// розрахуємо MD5-HMAC над Blowfish взламаним рядком
    DoInfo('MD5 розрахований над Blowfish взламаним рядком ', clGreen);
// використовуємо TProtection методи для побудови ланцюжка
    Protection := TCipher_Blowfish.Create('DEC Part I', nil);
    CodeString('Teststring', paCalc, fmtNONE); // fmtNONE = no Stringconvert

    DoInfo('CodeString()'#9+DigestStr(Format), clWindowText);
//---------------------------------------------------------------------------
    Protection := nil;
    Cipher := TCipher_Blowfish.Create('', nil);
    try
      // ініціалізуємо шифр та взламуємо рядок
      Cipher.InitKey('DEC Part I', nil);
      S := Cipher.EncodeString('Teststring');
      Cipher.Done;
```

```
      // розраховуємо MD5 на зашифрованим рядком
      Init;                       // ініціалізуємо MD5
      Calc(PChar(S)^, Length(S)); // розраховуємо MD5
      Done;                       // MD5
      // взламуємо MD5 повідомлення
      Cipher.EncodeBuffer(DigestKey^, DigestKey^, DigestKeySize);

      DoInfo('conventional'#9+DigestStr(Format), clWindowText);
    finally
      Cipher.Free;
    end;

// CodeBuffer(), CodeStream() та CodeFile()
    Free; // знищуємо MD5
  end;

end;

procedure TGForm.HashMACClick(Sender: TObject);
var
  S,FileName: String;
  MAC: TMAC;
  Protection: TProtection;
  HUser: THashClass;
begin
  M.Clear;

  HUser := THash_Haval192; // вибираємо хеш функцію для взламу
  FileName := ParamStr(0); // вибираємо файл для взламу

  DoInfo('Хеш повідомлення аутентифікаційного коду', clRed);
  DoInfo('Користувач визначає код як: ' + GetHashName(HUser), clMaroon);
  DoInfo('Простий тест на злам функції', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('SHA1:'#9 + sSelfTest[THash_SHA1.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//-----------------------------------------------------------------------------
  DoInfo('1. Generic THash_MD5(TMAC) -> MAC-MD5', clBlue);

  S := THash_MD5.CalcFile(FileName, TMAC.Create('DEC Part I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC Part I"'#9+S, clWindowText);

  S := THash_MD5.CalcFile(FileName, TMAC.Create('DEC PArt I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC PArt I"'#9+S, clWindowText);


//-----------------------------------------------------------------------------

  MAC := TMAC.Create('DEC', nil);
  try
    MAC.AddRef;  //

//-----------------------------------------------------------------------------
    DoInfo('2. Загальний TMAC -> використовує TMAC Instance,
THash_XXX(TMAC("DEC"))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
    DoInfo('3. Загальний TMAC -> використовує TMAC Instance with Protection,
THash_XXX(TMAC("DEC", TCipher_Blowfish("Secret")))', clBlue);
```

```
// визначіть  Blowfish Protection, these encrypt the final Hash.DigestKey
    MAC.Protection := TCipher_Blowfish.Create('Secret', nil);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  Визвольте Blowfish MAC Protection й визначіть  TRandom_LFSR protection
//  TRandom_LFSR has a Period from 2^400-1, see RNG.pas for more Details
    MAC.Protection := TRandom_LFSR.Create('Secret', 400, False, nil);

    DoInfo('4. Загальний TMAC -> використовує TMAC Instance with Protection,
THash_XXX(TMAC("DEC", TRandom_LFSR("Secret")))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  визвольте LFSR MAC Protection и визначіть  THash_MD4(TMAC) protection
// a Double HMAC -> HMAC-MD5-HMAC-MD4
    MAC.Protection := THash_MD4.Create(TMAC.Create('Secret', nil));
// Ланцюжок: THash_XXX -> TMAC('DEC') -> THash_MD4 -> TMAC('Secret')
    DoInfo('5. Загальний TMAC -> використовує TMAC Instance with Protection,
THash_XXX(TMAC("DEC", THash_MD4(TMAC("Secret"))))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
//  Change Password
    MAC.Protection.Protection := TMAC.Create('SecRet', nil);

    DoInfo('6. Загальний TMAC -> використовує TMAC Instance with Protection,
THash_XXX(TMAC("DEC", THash_MD4(TMAC("SecRet"))))', clBlue);

    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
// встановіть  MAC.Protection to THash_SHA1(TMAC("SecRet"))
// a HMAC-MD5-HMAC-SHA1
    MAC.Protection := THash_SHA1.Create(TMAC.Create('SecRet', nil));

    DoInfo('7. Загальний TMAC -> використовує TMAC Instance with Protection,
THash_XXX(TMAC("DEC", THash_SHA1(TMAC("SecRet"))))', clBlue);
```

```
    S := THash_MD5.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, MAC, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);

  finally
// реліз  MAC Instance та відлінкуйте Protections (THash_MD4 -> TMAC);
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('8. polymorph MAC -> THash_XXX(TCipher_Blowfish("Secret"))', clBlue);

  Protection := TCipher_Blowfish.Create('Secret', nil);
  try
    Protection.AddRef; // це загальний ресурс
    Protection.AddRef;
    // MD5-Blowfish-CTS-MAC
    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);
    // SHA1-Blowfish-CTS-MAC
    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release; // double AddRef -> double Release
    Protection.Release; // визвольте Cipher
  end;

//---------------------------------------------------------------------------
  DoInfo('9. поліморфичний MAC -> THash_XXX(TRandom_LFSR("Secret"))', clBlue);

  Protection := TRandom_LFSR.Create('Secret', 2032, False, nil); // Period
2^2032-1 see RNG.pas for Details
  try
    Protection.AddRef; // це загальний ресурс

    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);

    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release; // визвольте Random
  end;

//---------------------------------------------------------------------------
  DoInfo('10. поліморфичний MAC -> THash_XXX(TMAC("DEC"))', clBlue);
  DoInfo('Результати повинні бути такі ж як Step 2.', clMaroon);

  Protection := TMAC.Create('DEC', nil);
  try
    Protection.AddRef; // це загальний ресурс

    S := THash_MD5.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-MD5'#9+S, clWindowText);

    S := THash_SHA1.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-SHA1'#9+S, clWindowText);
```

```
    S := HUser.CalcFile(FileName, Protection, Format);
    DoInfo('MAC-HUser'#9+S, clWindowText);
  finally
    Protection.Release;
  end;

end;

procedure TGForm.MACwithRFC2104Click(Sender: TObject);

  function RepKey(Value, Count: Integer): String;
  begin
    SetLength(Result, Count);
    FillChar(PChar(Result)^, Count, Value);
  end;

var
  HUser: THashClass;
  MAC: TMAC;
  Data: array[1..50] of Byte;
  S: String;
  I: Integer;
  Stream: TMemoryStream;
begin
  M.Clear;

  HUser := DefaultHashClass;

  DoInfo('RFC2104 стандарт HMAC', clRed);
  DoInfo('Користувач визначає код як: ' + GetHashName(HUser), clMaroon);
  DoInfo(' MACs використовує Testcases з RFC2202, дивись Docus\RFC2202.html',
clMaroon);
  DoInfo('Це сірі значення з RFC2202.html', clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Hashs', clBlue);
  DoInfo('MD5:'#9 + sSelfTest[THash_MD5.SelfTest], clWindowText);
  DoInfo('SHA1:'#9 + sSelfTest[THash_SHA1.SelfTest], clWindowText);
  DoInfo('HUser:'#9 + sSelfTest[HUser.SelfTest], clWindowText);

//---------------------------------------------------------------------------
  DoInfo('Testcase No. 1', clBlue); // Test, використовує інший Key's для
кожного Thing

  S := THash_MD5.CalcString('Hi There', TMAC_RFC2104.Create(RepKey($0B, 16),
nil), fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'9294727a3638bb1c13f48ef8158bfc9d', clGrayText);

  S := THash_SHA1.CalcString('Hi There', TMAC_RFC2104.Create(RepKey($0B, 20),
nil), fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
  DoInfo(#9'b617318655057264e28bc0b6fb378c8ef146be00', clGrayText);

  S := HUser.CalcString('Hi There', TMAC_RFC2104.Create(RepKey($0B, 20), nil),
fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('Testcase No. 2', clBlue);

  MAC := TMAC_RFC2104.Create('Jefe', nil);
  try
    MAC.AddRef;

    S := THash_MD5.CalcString('what do ya want for nothing?', MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'750c783e6ab0b503eaa86e310a5db738', clGrayText);

    S := THash_SHA1.CalcString('what do ya want for nothing?', MAC, fmtHEXL);
```

```
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'effcdf6ae5eb2fa2d27416d5f184df9c259a7c79', clGrayText);

    S := HUser.CalcString('what do ya want for nothing?', MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 3', clBlue);

  FillChar(Data, SizeOf(Data), $DD);

  S := THash_MD5.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA,
16), nil), fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'56be34521d144c88dbb8c733f0e8b3f6', clGrayText);

  S := THash_SHA1.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA,
20), nil), fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
  DoInfo(#9'125d7342b9ac11cd91a39af48aa17b4f63f175d3', clGrayText);

  S := HUser.CalcBuffer(Data, SizeOf(Data), TMAC_RFC2104.Create(RepKey($AA, 20),
nil), fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 4', clBlue);

  FillChar(Data, SizeOf(Data), $CD);
  SetLength(S, 25);
  for I := 1 to 25 do Byte(S[I]) := I;

  MAC := TMAC_RFC2104.Create(S, nil);
  try
    MAC.AddRef;

    S := THash_MD5.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'697eaf0aca3a3aea3a75164746ffaa79', clGrayText);

    S := THash_SHA1.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'4c9007f4026250c6bc8414f9bf50c86c2d7235da', clGrayText);

    S := HUser.CalcBuffer(Data, SizeOf(Data), MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
  end;

//---------------------------------------------------------------------------
  DoInfo('Пакет тестування нейромережею № 5', clBlue);

  S := THash_MD5.CalcString('Test With Truncation',
TMAC_RFC2104.Create(RepKey($0C, 16), nil), fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'56461ef2342edc00f9bab995690efd4c', clGrayText);
  SetLength(S, 96 div 8 * 2);  // 96 Bits div 8 Bit * 2 Chars per Byte
  DoInfo('HMAC-MD5-96'#9+S, clWindowText);
  DoInfo(#9'56461ef2342edc00f9bab995', clGrayText);

  S := THash_SHA1.CalcString(«Тест з зкругленням»,
TMAC_RFC2104.Create(RepKey($0C, 20), nil), fmtHEXL);
  DoInfo('HMAC-SHA1'#9+S, clWindowText);
```

```
  DoInfo(#9'4c1a03424b55e07fe7f27be1d58bb9324a9a5a04', clGrayText);
  SetLength(S, 96 div 8 * 2);
  DoInfo('HMAC-SHA1-96'#9+S, clWindowText);
  DoInfo(#9'4c1a03424b55e07fe7f27be1', clGrayText);

  S := HUser.CalcString(«Тест з зкругленням», TMAC_RFC2104.Create(RepKey($0C,
20), nil), fmtHEXL);
  DoInfo('HMAC-HUser'#9+S, clWindowText);
  SetLength(S, 96 div 8 * 2);
  DoInfo('HMAC-HUser-96'#9+S, clWindowText);

//  Tests використовує Stream
  Stream := TMemoryStream.Create;
  MAC    := TMAC_RFC2104.Create(RepKey($AA, 80), nil);
  try
    MAC.AddRef;

//-----------------------------------------------------------------------------
    DoInfo('Пакет тестування нейромережею № 6', clBlue);

    Stream.Write('Test Using Larger Than Block-Size Key - Hash Key First', 54);
// не повинно використовуватися Stream.Position, використовується StreamSize = -
1, THash_XXX manage the Seeking
    S := THash_MD5.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'6b1ab7fe4bd7bf8f0b62e6ce61b9d0cd', clGrayText);

    S := THash_SHA1.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'aa4ae5e15272d00e95705637ce8a3b55ed402112', clGrayText);

    S := HUser.CalcStream(Stream, -1, MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

//-----------------------------------------------------------------------------
    DoInfo('Пакет тестування нейромережею № 7', clBlue);

    Stream.Size := 0;
    Stream.Write('Тест нейронною мережю використовує Larger Than Block-Size Key
and Larger Than One Block-Size Data', 73);
// Нейронною мережею встановлюється Stream.Position, we використовує a
StreamSize = Stream.Size
    Stream.Position := 0;
    S := THash_MD5.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'6f630fad67cda0ee1fb1f562db3aa53e', clGrayText);

    Stream.Position := 0;
    S := THash_SHA1.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'e8e99d0f45237d786d6bbaa7965c7808bbff1a91', clGrayText);

    Stream.Position := 0;
    S := HUser.CalcStream(Stream, Stream.Size, MAC, fmtHEXL);
    DoInfo('HMAC-HUser'#9+S, clWindowText);

  finally
    MAC.Release;
    Stream.Free;
  end;

end;

procedure TGForm.ViewRFC2202html1Click(Sender: TObject);
var
  S: String;
begin
  S := ExtractFilePath(ParamStr(0));
  SetLength(S, Length(S)-1);
```

```
  S := ExtractFilePath(S) + 'Docus\RFC2202.html';
  ShellExecute(Handle, nil, PChar(S), nil, nil, sw_ShowNormal);
end;

procedure TGForm.TransactionNumbersTANs1Click(Sender: TObject);
const
  HashTAN : THashClass = THash_SHA1;
  maxTANEntries = 10; // TAN List have 10 Numbers

// робить короткий рядок
  function FoldStr(const Value: String): String;
  const
    maxLen = 8; // робить не більш коротке чим 6, 6 байт – це тільки 2^48
комбінацій !
  var
    I,Len: Integer;
  begin
    Result := Value;
    Len := Length(Result);
    for I := 1 to Len do
      Byte(Result[I]) := Byte(Result[I]) xor Byte(Result[(I + maxLen) mod Len]);
    SetLength(Result, maxLen);
  end;

// Складає Ліст шифрів для клієнта
  function CreateTANList(const SeedTANList, SeedTAN, Name: String; ID: Integer;
var LastTAN: String): TStringList;
  type
    PClient = ^TClient;
    TClient = packed record
                Name: array[0..80] of Char; // Імя клієнта
                ID: Integer;                // ID клієнта
                Seed: array[0..64] of Char;
                TANCount: Integer;          // лічильник TAN lists
              end;
   var
     Client: TClient;
     S: String;
     I: Integer;
   begin
// складаємо лист шифрів
     Result := TStringList.Create;
// устанавлюємо Client Infos
     FillChar(Client, Sizeof(Client), 0);
     StrPLCopy(Client.Name, AnsiUpperCase(Trim(Name)), SizeOf(Client.Name));
     Client.ID := ID;
     Client.TANCount := 1;

// Розраховуються безпечні параметри клієнта з параметрів серверу S0
     S := FormatToStr(PChar(SeedTANList), -1, Format);
     I := ID;
     repeat
       S := HashTAN.CalcString(S, nil, fmtCOPY);
       Dec(I);
     until I <= 0;
     StrPLCopy(Client.Seed, S, SizeOf(Client.Seed));
     S := HashTAN.CalcBuffer(Client, SizeOf(Client), nil, fmtCOPY);
     I := maxTANEntries;
     repeat
       S := HashTAN.CalcString(S, nil, fmtCOPY);
       S := FoldStr(S);
       Result.Insert(0, StrToFormat(PChar(S), Length(S), Format));
       Dec(I);
     until I <= 0;
     S := HashTAN.CalcString(S, nil, fmtCOPY);
     S := FoldStr(S);
     S := S + FormatToStr(PChar(SeedTAN), -1, Format);
     LastTAN := HashTAN.CalcString(S, nil, Format);
   end;
```

```
// перевіряємо CurrentTAN з LastTAN/SeedTAN та записуємо наступний LastTAN
   function CheckTAN(const SeedTAN, LastTAN: String; var CurrentTAN: String):
Boolean;
   var
     C,L,S: String;
     I: Integer;
   begin
     try
       S := FormatToStr(PChar(SeedTAN), -1, Format);
       C := FormatToStr(PChar(CurrentTAN), -1, Format);
       L := FormatToStr(PChar(LastTAN), -1, Format);
       I := maxTANEntries; // max. TAN List Count
       repeat
         C := HashTAN.CalcString(C, nil, fmtCOPY);
         C := FoldStr(C);
// розраховуємо коректний TAN
         Result := HashTAN.CalcString(C + S, nil, fmtCOPY) = L;
         Dec(I);
       until Result or (I <= 0);
       C := FormatToStr(PChar(CurrentTAN), -1, Format) + S;
       CurrentTAN := HashTAN.CalcString(C, nil, Format);
     except
       Result := False;
       Application.HandleException(nil);
     end;
   end;

var
  SeedTANList, SeedTAN: String;
  TANList: TStringList;
  I: Integer;
  LastTAN: String;
  TAN: String;
begin
  M.Clear;
  DoInfo('Кількість транзакцій для визначення паролю ', clRed);
  DoInfo('Hash алгоритм це: ' + GetHashName(HashTAN), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Hashs', clBlue);
  DoInfo('HashTAN:'#9 + sSelfTest[HashTAN.SelfTest], clWindowText);
  DoInfo('будуємо сервер S0', clBlue);

  SeedTANList := HashTAN.CalcString('Пароль серверу "Sample BANK of Ukraine"',
nil, Format);
  SeedTAN     := SeedTANList; //

  DoInfo('S0:'#9+SeedTANList, clWindowText);

  DoInfo('будуємо TAN list для "Matvienko Tatiyana"', clBlue);

  TANList := CreateTANList(SeedTANList, SeedTAN, 'Matvienko Tatiyana', 54,
LastTAN);

  try
// На сервері нейронної мережі побудована база даних з полями:
// ClientID та Last використовують TAN
// запам’ятовуємо перший TAN (LastTAN) в базі даних нейронної мережі

// Для Клієнта
    DoInfo('TAN список клієнта:', clWindowText);
    for I := 0 to TANList.Count-1 do
      DoInfo(IntToStr(I)+ ':'#9+TANList[I], clWindowText);

    DoInfo('Нейронна мережа зробила транзакцію', clBlue);

// 1. TA ----------------------------------------------------------------
    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('Current TAN:'#9 + TAN, clWindowText);
```

```
// Clients записується  TAN та Client ID надається в сервер нейронної мережі
// Server шукає в Database Client ID, достає LastTAN та
// перевіряє TAN
    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN is ok', clGreen);
// зберігаємо поточний TAN в Database та останній клієнтський TAN
      LastTAN := TAN;
      DoInfo('останній TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN плохий', clMaroon);
    end;
    DoInfo('', clWindowText);

// 2. TA ---------------------------------------------------------------------
    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('поточний TAN:'#9 + TAN, clWindowText);
    Delete(TAN, 1, 4);
    DoInfo('Плохий TAN:'#9 + TAN, clMaroon);
// on the Server, check the TAN
    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN нормальний', clGreen);
      LastTAN := TAN;
      DoInfo('Останній TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;
    DoInfo('', clWindowText);

// 3. TA ---------------------------------------------------------------------
    TANList.Delete(0); TANList.Delete(0);

    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('Current TAN:'#9 + TAN, clWindowText);

/    if CheckTAN(SeedTAN, LastTAN, TAN) then
    begin
      DoInfo('TAN is ok', clGreen);
// Зберігаємо поточний TAN в Database використовуємо останній TAN
      LastTAN := TAN;
      DoInfo('Last TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;

  finally
    TANList.Free;
  end;
end;

procedure TGForm.UsingfromCiphers1Click(Sender: TObject);
const
  sMode: array[TCipherMode] of String = ('cmCTS', 'cmCBC', 'cmCFB', 'cmOFB',
                                         'cmECB', 'cmCTSMAC', 'cmCBCMAC',
'cmCFBMAC');
var
  CUser: TCipherClass;
  Buffer: array[0..15] of Byte;
  I: Integer;
  S: String;
  Stream: TMemoryStream;
  K: TCipherMode;
begin
  M.Clear;
  CUser := TCipher_Blowfish; // вибираємо шифр для взламу
```

```
  for I := Low(Buffer) to High(Buffer) do Buffer[I] := I + 32; // setup Buffer

  DoInfo('Шифр обрано', clRed);
  DoInfo('Користувач визначив шифр як : ' + GetCipherName(CUser), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Ciphers', clBlue);
  DoInfo('CUser:'#9 + sSelfTest[CUser.SelfTest], clWindowText);
  DoInfo('Blowfish:'#9 + sSelfTest[TCipher_Blowfish.SelfTest], clWindowText);
  DoInfo('IDEA:'#9 + sSelfTest[TCipher_IDEA.SelfTest], clWindowText);
  DoInfo('GOST:'#9 + sSelfTest[TCipher_GOST.SelfTest], clWindowText);


  with CUser.Create('', nil) do
  try
//-----------------------------------------------------------------------------
    DoInfo('1. Шифрування/дешифрування файлу, ParamStr(0), DEMO.EXE', clBlue);

    InitKey('DEC', nil);
    EncodeFile(ParamStr(0), ChangeFileExt(ParamStr(0), '.ENC'));
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Done;
//    InitKey('DEC', nil);

    DecodeFile(ChangeFileExt(ParamStr(0), '.ENC'), ChangeFileExt(ParamStr(0),
'.DEC'));
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect; //

//-----------------------------------------------------------------------------
    DoInfo('2. Шифрування/дешифрування рядка, "Тест шифрувння рядка"', clBlue);
    InitKey('DEC Part I', nil);

    S := EncodeString('Тест дешифрувння рядка');
    DoInfo('Encryted:'#9+ StrToFormat(PChar(S), Length(S), Format),
clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
    Done;

    S := DecodeString(S);

    DoInfo('Decryted:'#9+ S, clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect;

//-----------------------------------------------------------------------------
    DoInfo('3. Шифрування/дешифрування буфера, "' +StrToFormat(@Buffer,
Sizeof(Buffer), Format) + '"', clBlue);
    InitKey('DEC Part I', nil);

    EncodeBuffer(Buffer, Buffer, SizeOf(Buffer));
    DoInfo('Шифрування:'#9+ StrToFormat(@Buffer, Sizeof(Buffer), Format),
clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
    Done;

    DecodeBuffer(Buffer, Buffer, SizeOf(Buffer));

    DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(@Buffer, Sizeof(Buffer),
Format), clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect;

//-----------------------------------------------------------------------------
    DoInfo('4. Шифрування/дешифрування потоку, "Partial Stream En/Decryption"',
clBlue);
```

```
    InitKey('DEC Part I', nil);

    Stream := TMemoryStream.Create;
    try
      S := 'Partial Stream En/Decryption';
      Stream.Write(PChar(S)^, Length(S));

      Stream.Position := 8;
      EncodeStream(Stream, Stream, 6);

      DoInfo('Шифрування:'#9+ StrToFormat(Stream.Memory, Stream.Size, Format),
clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
      Done;

      Stream.Position := 8;
      DecodeStream(Stream, Stream, 6);

      DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(Stream.Memory,
Stream.Size, Format), clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    finally
      Stream.Free;
    end;

//---------------------------------------------------------------------------
    DoInfo('5. Різні режими шифрування', clBlue);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      InitKey('DEC', nil);
      S := EncodeString('Тест нейронною мережею зашифрованого рядка');
      DoInfo('Шифрування:'#9+ StrToFormat(PChar(S), Length(S), Format),
clWindowText);

      Done;

      S := DecodeString(S);
      DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(PChar(S), Length(S),
Format), clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('6. Використовувати TProtection-Method CodeString() with any
Protection', clBlue);

  with CUser.Create('', nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('Без шифрування', clBlue);

    InitKey('DEC', nil);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
```

```
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено  TRandom_LFSR("DEC")', clBlue);
    Protection := TRandom_LFSR.Create('DEC', 400, False, nil);
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено THash_MD5(TMAC_RFC2104("DEC"))', clBlue);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('DEC', nil));
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
    end;

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('7. Використовуємо TProtections Methods', clBlue);

  with CUser.Create('', nil) do
  try
    HashClass := THash_MD5;  // установлюємо інші HashClass for the InitKey()
    InitKey('DEC', nil);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paEncode/paDecode)', clGreen);

    S := CodeString('CodeString()', paEncode, Format);
    DoInfo('Шифрування:'#9+ S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paScramble)', clGreen);

    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paWipe)', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
```

```
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);

  finally
    Free;
  end;
//-------------------------------------------------------------------------
  DoInfo('8. A secure зашифрований ', clBlue);
//  demonstrate a Multi-En/Decryption that використовує 3 Ciphers in a Chain.

  with TCipher_Blowfish.Create('DEC',
         TCipher_IDEA.Create('DEC',
           TCipher_GOST.Create('DEC',
             TRandom_LFSR.Create('Scramble', 128, False, nil)))) do
  try
    S := CodeString('Добрий DEC Part I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

procedure TGForm.CipherMACClick(Sender: TObject);
const
  sMode: array[TCipherMode] of String = ('CTS-MAC', 'CBC-MAC', 'CFB-MAC',
                                         'invalid', 'invalid',
                                         'CTS-MAC', 'CBC-MAC', 'CFB-MAC');
var
  CUser: TCipherClass;
  K: TCipherMode;
  I: Integer;
begin
  M.Clear;
  CUser := TCipher_Blowfish; // вибираємо шифр для взламу

  DoInfo('Посилається автентифікаційний код з шифром', clRed);
  DoInfo('Користувач визначив шифр як : ' + GetCipherName(CUser), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Ciphers', clBlue);
  DoInfo('CUser:'#9 + sSelfTest[CUser.SelfTest], clWindowText);
  DoInfo('SCOP:'#9 + sSelfTest[TCipher_SCOP.SelfTest], clWindowText);

//-------------------------------------------------------------------------
  DoInfo('1. MAC для ParamStr(0), DEMO.EXE', clBlue);
  with CUser.Create('DEC', nil) do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() in MAC Mode: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() in MAC Mode: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
```

```
  end;

//---------------------------------------------------------------------------
  DoInfo('2. MAC для ParamStr(0), DEMO.EXE Захищено Blowfish("Secret")',
clBlue);
  with CUser.Create('DEC', TCipher_Blowfish.Create('Secret', nil)) do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() in MAC Mode: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() in MAC Mode: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('3. MAC''s with TProtection Method CodeString', clBlue);

  with CUser.Create('DEC', nil) do
  try
// визначіть  HMAC-MD5-LFSR128-SCOP protection :-)
//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for MD5',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('SecRet 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for LFSR',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('SecRet 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
```

```
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for SCOP',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('SecRet 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;
  finally
    Free;
  end;
 CodeBuffer(), CodeString()
//   CodeStream(), CodeFile()
// - with Action = paCalc it's the Result from CodeString()
end;

procedure TGForm.UsingfromRandoms1Click(Sender: TObject);
var
  I: Integer;
  S,SaveState: String;
  Buf: array[0..15] of Byte;
begin
  M.Clear;

  DoInfo('Random using', clRed);
//---------------------------------------------------------------------------
  DoInfo('1. TRandom_LFSR Instance with Period 2^400-1', clBlue);

  with TRandom_LFSR.Create('', 400, False, nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC Part I"', clBlue);
    Seed('DEC Part I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC ParT I"', clBlue);
    Seed('DEC ParT I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, default Seed', clBlue);
    Seed('', 0);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, randomized Seed', clBlue);
    Seed('', -1);
```

```
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1000 to 1000, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('Change Period to 2^2032-1', clGreen);
    Size := 2032;

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1 to 1, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed', clBlue);
    Seed('', 0);

    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed with saveing of State', clBlue);
    DoInfo('Switch back to Period 2^128-1', clGreen);
    Size := 128;
    Protection := TCipher_Blowfish.Create('DEC', nil);
    Seed('', -1);          SaveState := State;
    DoInfo('SaveState:' + InsertBlocks(SaveState, #9, #10, 64), clGreen);
// випадковий Buffer
    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, SizeOf(Buf), Format), clWindowText);

    Seed('', -1); //
    DoInfo('Стан відновлено з SaveState', clGreen);

    State := SaveState;
    SetLength(S, Sizeof(Buf));
    Buffer(PChar(S)^, Length(S));

    DoInfo(StrToFormat(PChar(S), Length(S), Format), clWindowText);

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('2. Глобальна випадкова змінна "RND"', clBlue);
// RND is per default TRandom_LFSR('', 128);
  RND.Seed('', 0);
  RND.Buffer(Buf, SizeOf(Buf));
  DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
```

```
  DoInfo('3. TProtection методи з TRandom_LFSR', clBlue);
  with TRandom_LFSR.Create('DEC', 128, False, nil) do
  try
//-------------------------------------------------------------------------------
    DoInfo('Кодування / декодування нейронною мережею з TRandom', clGreen);

    S := CodeString('Добрий DEC Part I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);

//-------------------------------------------------------------------------------
    DoInfo('Утаємничення з TRandom', clGreen);

    S := CodeString('Добрий DEC Part I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

    DoInfo('BasicSeed changed from: '+ SysUtils.Format('$%0.8x to $%0.8x',
[BasicSeed, BasicSeed +1]), clGreen);
    BasicSeed := BasicSeed +1; // використовує інший BasicSeed
    S := CodeString('Добрий DEC Part I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

//-------------------------------------------------------------------------------
    DoInfo('Взломано з TRandom', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
    S := CodeString('Добрий DEC Part I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Part I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Part I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

end.
```

Chiper.pas – файл реалізації алгоритмів для крипто аналізу нейронною мережею

```
unit Cipher;

interface

{$I VER.INC}

uses SysUtils, Classes, DECUtil, Hash;

const {Коди помилок}
  errGeneric        = 0;  {Помилка генерації}
  errInvalidKey     = 1;  {Ключ декодування некоректний}
  errInvalidKeySize = 2;  {Розмір ключа дуже великий}
  errNotInitialized = 3;  {Методи Init() або InitKey() не викликаються}
  errInvalidMACMode = 4;  {CalcMAC не повертає  cmECB, cmOFB}
  errCantCalc       = 5;

type
  ECipherException = class(Exception)
  public
    ErrorCode: Integer;
  end;

{Перелік алгоритмів для крипто аналізу у вигляді классів}
  TCipher_Gost         = class;
  TCipher_Blowfish     = class;
  TCipher_IDEA         = class;
  TCipher_SAFER        = class;
  TCipher_SAFER_K40    = class;
  TCipher_SAFER_SK40   = class;
  TCipher_SAFER_K64    = class;
  TCipher_SAFER_SK64   = class;
  TCipher_SAFER_K128   = class;
  TCipher_SAFER_SK128  = class;
  TCipher_TEA          = class;
  TCipher_TEAN         = class;
  TCipher_SCOP         = class;
  TCipher_Q128         = class;
  TCipher_3Way         = class;
  TCipher_Twofish      = class;
  TCipher_Shark        = class;
  TCipher_Square       = class;

  TCipherMode = (cmCTS, cmCBC, cmCFB, cmOFB, cmECB, cmCTSMAC, cmCBCMAC,
cmCFBMAC);
{ Режими шифрування:
  cmCTS     Cipher Text Stealing
  cmCBC     Cipher Block Chaining
  cmCFB     K-bit Cipher Feedback
  cmOFB     K-bit Output Feedback
  cmECB *   Electronic Codebook

  cmCTSMAC  Message Authentication Code в режимі cmCTS
  cmCBCMAC – CBC-MAC
  cmCFBMAC –  CFB-MAC
}

  TCipherClass = class of TCipher;

  TCipher = class(TProtection)
  private
    FMode: TCipherMode;
    FHash: THash;
    FHashClass: THashClass;
    FKeySize: Integer;
    FBufSize: Integer;
    FUserSize: Integer;
```

```
    FBuffer: Pointer;
    FVector: Pointer;
    FFeedback: Pointer;
    FUser: Pointer;
    FFlags: Integer;
    function GetHash: THash;
    procedure SetHashClass(Value: THashClass);
    procedure InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
    procedure InternalCodeFile(const Source, Dest: String; Encode: Boolean);
  protected
    function GetFlag(Index: Integer): Boolean;
    procedure SetFlag(Index: Integer; Value: Boolean); virtual;
{використовуються в методі Init()}
    procedure InitBegin(var Size: Integer);
    procedure InitEnd(IVector: Pointer); virtual;
{ анульовано}
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
virtual;
    class function TestVector: Pointer; virtual;
{анульовано TProtection Methods}
    procedure CodeInit(Action: TPAction); invalidated;
    procedure CodeDone(Action: TPAction); invalidated;
    procedure CodeBuf(var Buffer; const BufferSize: Integer; Action: TPAction);
invalidated;
{ анульовано}
    procedure Encode(Data: Pointer); virtual;
{після декодування функції анульовано}
    procedure Decode(Data: Pointer); virtual;
    property User: Pointer read FUser;
    property Buffer: Pointer read FBuffer;
    property UserSize: Integer read FUserSize;
  public
    constructor Create(const Password: String; AProtection: TProtection);
    destructor Destroy; invalidated;
    class function MaxKeySize: Integer;
{тест на коректність роботи}
    class function SelfTest: Boolean;
{ініціалізація форм для шифрування}
    procedure Init(const Key; Size: Integer; IVector: Pointer); virtual;
    procedure InitKey(const Key: String; IVector: Pointer);
    procedure Done; virtual;
    procedure Protect; virtual;

    procedure EncodeBuffer(const Source; var Dest; DataSize: Integer);
    procedure DecodeBuffer(const Source; var Dest; DataSize: Integer);
    function  EncodeString(const Source: String): String;
    function  DecodeString(const Source: String): String;
    procedure EncodeFile(const Source, Dest: String);
    procedure DecodeFile(const Source, Dest: String);
    procedure EncodeStream(const Source, Dest: TStream; DataSize: Integer);
    procedure DecodeStream(const Source, Dest: TStream; DataSize: Integer);

    function CalcMAC(Format: Integer): String;

{Cipher Mode = cmXXX}
    property Mode: TCipherMode read FMode write FMode;
{ поточний Hash-Object, будує Digest з InitKey()}
    property Hash: THash read GetHash;
{ Class Hash-Object}
    property HashClass: THashClass read FHashClass write SetHashClass;
{максимальний розмір ключа та буфера }
    property KeySize: Integer read FKeySize;
    property BufSize: Integer read FBufSize;

{Init() повинно визиватися}
    property Initialized: Boolean index 1 read GetFlag write SetFlag;
    property Vector: Pointer read FVector;
    property Feedback: Pointer read FFeedback;
```

```
    property HasHashKey: Boolean index 0 read GetFlag;
  end;

// Опис шифрів

  TCipher_Gost = class(TCipher) {російський шифр}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Blowfish = class(TCipher)
  private
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
    procedure Encode386(Data: Pointer);
    procedure Decode386(Data: Pointer);
  {$ENDIF}
{$ENDIF}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_IDEA = class(TCipher) {International Data Encryption Algorithm }
  private
    procedure Cipher(Data, Key: PWordArray);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TSAFERMode = (smDefault, smK40, smK64, smK128, smStrong, smSK40, smSK64,
smSK128);

  TCipher_SAFER = class(TCipher)
  private
    FRounds: Integer;
    FSAFERMode: TSAFERMode;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure InitNew(const Key; Size: Integer; IVector: Pointer; SAFERMode:
TSAFERMode);
    property Rounds: Integer read FRounds write SetRounds;
  end;

  TCipher_SAFER_K40 = class(TCipher_SAFER)
```

```
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK40 = class(TCipher_SAFER_K40)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K64 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK64 = class(TCipher_SAFER_K64)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K128 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK128 = class(TCipher_SAFER_K128)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_TEA = class(TCipher) {Tiny Encryption Algorithm}
  private
    FRounds: Integer;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    property Rounds: Integer read FRounds write SetRounds;
  end;

  TCipher_TEAN = class(TCipher_TEA) {Tiny Encryption Algorithm, extended
Version}
  protected
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  end;
```

```
  TCipher_SCOP = class(TCipher) {Stream Cipher in Blockmode}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure Done; invalidated;
  end;

  TCipher_Q128 = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_3Way = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Twofish = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Shark = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Square = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

function DefaultCipherClass: TCipherClass;
```

```
procedure SetDefaultCipherClass(CipherClass: TCipherClass);
procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
function UnregisterCipher(const ACipher: TCipherClass): Boolean;
function CipherList: TStrings;
procedure CipherNames(List: TStrings);
function GetCipherClass(const Name: String): TCipherClass;
function GetCipherName(CipherClass: TCipherClass): String;

const
  CheckCipherKeySize: Boolean = False;

implementation

uses DECConst, Windows;

{$I *.inc}
{$I Square.inc}

const
  FDefaultCipherClass : TCipherClass = TCipher_Blowfish;
  FCipherList         : TStringList  = nil;

function DefaultCipherClass: TCipherClass;
begin
  Result := FDefaultCipherClass;
end;

procedure SetDefaultCipherClass(CipherClass: TCipherClass);
begin
  if CipherClass = nil then FDefaultCipherClass := TCipher_Blowfish
    else FDefaultCipherClass := CipherClass;
end;

procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
var
  E: ECipherException;
begin
  E := ECipherException.Create(Msg);
  E.ErrorCode := ErrorCode;
  raise E;
end;

function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
var
  I: Integer;
  S: String;
begin
  Result := False;
  if ACipher = nil then Exit;
  S := Trim(AName);
  if S = '' then
  begin
    S := ACipher.ClassName;
    if S[1] = 'T' then Delete(S, 1, 1);
    I := Pos('_', S);
    if I > 0 then Delete(S, 1, I);
  end;
  S := S + '=' + ADescription;
  I := CipherList.IndexOfObject(Pointer(ACipher));
  if I < 0 then CipherList.AddObject(S, Pointer(ACipher))
    else CipherList[I] := S;
  Result := True;
end;

function UnregisterCipher(const ACipher: TCipherClass): Boolean;
var
```

```
  I: Integer;
begin
  Result := False;
  repeat
    I := CipherList.IndexOfObject(Pointer(ACipher));
    if I < 0 then Break;
    Result := True;
    CipherList.Delete(I);
  until False;
end;

function CipherList: TStrings;
begin
  if not IsObject(FCipherList, TStringList) then FCipherList :=
TStringList.Create;
  Result := FCipherList;
end;

procedure CipherNames(List: TStrings);
var
  I: Integer;
begin
  if not IsObject(List, TStrings) then Exit;
  for I := 0 to CipherList.Count-1 do
    List.AddObject(FCipherList.Names[I], FCipherList.Objects[I]);
end;

function GetCipherClass(const Name: String): TCipherClass;
var
  I: Integer;
  N: String;
begin
  Result := nil;
  N := Name;
  I := Pos('_', N);
  if I > 0 then Delete(N, 1, I);
  for I := 0 to CipherList.Count-1 do
    if AnsiCompareText(N, GetShortClassName(TClass(FCipherList.Objects[I]))) = 0
then
    begin
      Result := TCipherClass(FCipherList.Objects[I]);
      Exit;
    end;
  I := FCipherList.IndexOfName(N);
  if I >= 0 then Result := TCipherClass(FCipherList.Objects[I]);
end;

function GetCipherName(CipherClass: TCipherClass): String;
var
  I: Integer;
begin
  I := CipherList.IndexOfObject(Pointer(CipherClass));
  if I >= 0 then Result := FCipherList.Names[I]
    else Result := GetShortClassName(CipherClass);
end;

function TCipher.GetFlag(Index: Integer): Boolean;
begin
  Result := FFlags and (1 shl Index) <> 0;
end;

procedure TCipher.SetFlag(Index: Integer; Value: Boolean);
begin
  Index := 1 shl Index;
  if Value then FFlags := FFlags or Index
    else FFlags := FFlags and not Index;
end;

procedure TCipher.InitBegin(var Size: Integer);
```

```
begin
  Initialized := False;
  Protect;
  if Size < 0 then Size := 0;
  if Size > KeySize then
    if not CheckCipherKeySize then Size := KeySize
      else RaiseCipherException(errInvalidKeySize, Format(sInvalidKeySize,
[ClassName, 0, KeySize]));
end;

procedure TCipher.InitEnd(IVector: Pointer);
begin
  if IVector = nil then Encode(Vector)
    else Move(IVector^, Vector^, BufSize);
  Move(Vector^, Feedback^, BufSize);
  Initialized := True;
end;

class procedure TCipher.GetContext(var ABufSize, AKeySize, AUserSize: Integer);
begin
  ABufSize := 0;
  AKeySize := 0;
  AUserSize := 0;
end;

class function TCipher.TestVector: Pointer;
begin
  Result := GetTestVector;
end;

procedure TCipher.Encode(Data: Pointer);
begin
end;

procedure TCipher.Decode(Data: Pointer);
begin
end;

constructor TCipher.Create(const Password: String; AProtection: TProtection);
begin
  inherited Create(AProtection);
  FHashClass := DefaultHashClass;
  GetContext(FBufSize, FKeySize, FUserSize);
  GetMem(FVector, FBufSize);
  GetMem(FFeedback, FBufSize);
  GetMem(FBuffer, FBufSize);
  GetMem(FUser, FUserSize);
  Protect;
  if Password <> '' then InitKey(Password, nil);
end;

destructor TCipher.Destroy;
begin
  Protect;
  ReallocMem(FVector, 0);
  ReallocMem(FFeedback, 0);
  ReallocMem(FBuffer, 0);
  ReallocMem(FUser, 0);
  FHash.Release;
  FHash := nil;
  inherited Destroy;
end;

class function TCipher.MaxKeySize: Integer;
var
  Dummy: Integer;
begin
  GetContext(Dummy, Result, Dummy);
end;
```

```
class function TCipher.SelfTest: Boolean;
var
  Data: array[0..63] of Char;
  Key: String;
  SaveKeyCheck: Boolean;
begin
  Result       := InitTestIsOk;
  Key          := ClassName;
  SaveKeyCheck := CheckCipherKeySize;
  with Self.Create('', nil) do
  try
    CheckCipherKeySize := False;
    Mode := cmCTS;
    Init(PChar(Key)^, Length(Key), nil);
    EncodeBuffer(GetTestVector^, Data, 32);
    Result := Result and (MemCompare(TestVector, @Data, 32) = 0);
    Done;
    DecodeBuffer(Data, Data, 32);
    Result := Result and (MemCompare(GetTestVector, @Data, 32) = 0);
  finally
    CheckCipherKeySize := SaveKeyCheck;
    Free;
  end;
  FillChar(Data, SizeOf(Data), 0);
end;

procedure TCipher.Init(const Key; Size: Integer; IVector: Pointer);
begin
end;

procedure TCipher.InitKey(const Key: String; IVector: Pointer);
var
  I: Integer;
begin
  Hash.Init;
  Hash.Calc(PChar(Key)^, Length(Key));
  Hash.Done;
  I := Hash.DigestKeySize;
  if I > FKeySize then I := FKeySize;
  Init(Hash.DigestKey^, I, IVector);
  EncodeBuffer(Hash.DigestKey^, Hash.DigestKey^, Hash.DigestKeySize);
  Done;
  SetFlag(0, True);
end;

procedure TCipher.Done;
begin
  if MemCompare(FVector, FFeedback, FBufSize) = 0 then Exit;
  Move(FFeedback^, FBuffer^, FBufSize);
  Move(FVector^, FFeedback^, FBufSize);
end;

procedure TCipher.Protect;
begin
  SetFlag(0, False);
  Initialized := False;
// a Crypto Fanatican say: this is better !!
  FillChar(FVector^, FBufSize, $AA);
  FillChar(FFeedback^, FBufSize, $AA);
  FillChar(FBuffer^, FBufSize, $AA);
  FillChar(FUser^, FUserSize, $AA);

  FillChar(FVector^, FBufSize, $55);
  FillChar(FFeedback^, FBufSize, $55);
  FillChar(FBuffer^, FBufSize, $55);
  FillChar(FUser^, FUserSize, $55);

  FillChar(FVector^, FBufSize, $FF);
```

```
  FillChar(FFeedback^, FBufSize, $FF);
  FillChar(FBuffer^, FBufSize, 0);
  FillChar(FUser^, FUserSize, 0);
end;

function TCipher.GetHash: THash;
begin
  if not IsObject(FHash, THash) then
  begin
    if FHashClass = nil then FHashClass := DefaultHashClass;
    FHash := FHashClass.Create(nil);
    FHash.AddRef;
  end;
  Result := FHash;
end;

procedure TCipher.SetHashClass(Value: THashClass);
begin
  if Value <> FHashClass then
  begin
    FHash.Release;
    FHash := nil;
    FHashClass := Value;
    if FHashClass = nil then FHashClass := DefaultHashClass;
  end;
end;

procedure TCipher.InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
const
  maxBufSize = 1024 * 4;
var
  Buf: PChar;
  SPos: Integer;
  DPos: Integer;
  Len: Integer;
  Proc: procedure(const Source; var Dest; DataSize: Integer) of object;
  Size: Integer;
begin
  if Source = nil then Exit;
  if Encode or (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) then Proc :=
EncodeBuffer
    else Proc := DecodeBuffer;
  if Dest = nil then Dest := Source;
  if DataSize < 0 then
  begin
    DataSize := Source.Size;
    Source.Position := 0;
  end;
  Buf := nil;
  Size := DataSize;
  DoProgress(Self, 0, Size);
  try
    Buf    := AllocMem(maxBufSize);
    DPos   := Dest.Position;
    SPos   := Source.Position;
    if Mode in [cmCTSMAC, cmCBCMAC, cmCFBMAC] then
    begin
      while DataSize > 0 do
      begin
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
    end else
```

```
      while DataSize > 0 do
      begin
        Source.Position := SPos;
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        SPos := Source.Position;
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
        Dest.Position := DPos;
        Dest.Write(Buf^, Len);
        DPos := Dest.Position;
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
  finally
    DoProgress(Self, 0, 0);
    ReallocMem(Buf, 0);
  end;
end;

procedure TCipher.InternalCodeFile(const Source, Dest: String; Encode: Boolean);
var
  S,D: TFileStream;
begin
  S := nil;
  D := nil;
  try
    if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then
    begin
      S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
      D := S;
    end else
      if (AnsiCompareText(Source, Dest) <> 0) and (Trim(Dest) <> '') then
      begin
        S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
        D := TFileStream.Create(Dest, fmCreate);
      end else
      begin
        S := TFileStream.Create(Source, fmOpenReadWrite);
        D := S;
      end;
    InternalCodeStream(S, D, -1, Encode);
  finally
    S.Free;
    if S <> D then
    begin
{$IFDEF VER_D3H}
      D.Size := D.Position;
{$ENDIF}
      D.Free;
    end;
  end;
end;

procedure TCipher.EncodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, True);
end;

procedure TCipher.DecodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, False);
end;

procedure TCipher.EncodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, True);
end;
```

```
procedure TCipher.DecodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, False);
end;

function TCipher.EncodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  EncodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

function TCipher.DecodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  DecodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

procedure TCipher.EncodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Encode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        while DataSize >= FBufSize do
        begin
          XORBuffers(S, FFeedback, FBufSize, D);
          Encode(D);
          XORBuffers(D, FFeedback, FBufSize, FFeedback);
          Inc(S, FBufSize);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(S, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCBC:
      begin
        F := FFeedback;
        while DataSize >= FBufSize do
```

```
      begin
        XORBuffers(S, F, FBufSize, D);
        Encode(D);
        F := D;
        Inc(S, FBufSize);
        Inc(D, FBufSize);
        Dec(DataSize, FBufSize);
      end;
      Move(F^, FFeedback^, FBufSize);
      if DataSize > 0 then
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        XORBuffers(S, FBuffer, DataSize, D);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
      end;
    end;
  cmCFB:
    while DataSize > 0 do
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      D^ := S^ xor PByte(FBuffer)^;
      Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
      PByteArray(FFeedback)[FBufSize-1] := D^;
      Inc(D);
      Inc(S);
      Dec(DataSize);
    end;
  cmOFB:
    while DataSize > 0 do
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      D^ := S^ xor PByte(FBuffer)^;
      Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
      PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
      Inc(D);
      Inc(S);
      Dec(DataSize);
    end;
  cmCTSMAC:
    begin
      while DataSize >= FBufSize do
      begin
        XORBuffers(S, FFeedback, FBufSize, FBuffer);
        Encode(FBuffer);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        Inc(S, FBufSize);
        Dec(DataSize, FBufSize);
      end;
      if DataSize > 0 then
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
      end;
    end;
  cmCBCMAC:
    begin
      while DataSize >= FBufSize do
      begin
        XORBuffers(S, FFeedback, FBufSize, FBuffer);
        Encode(FBuffer);
        Move(FBuffer^, FFeedback^, FBufSize);
        Inc(S, FBufSize);
        Dec(DataSize, FBufSize);
      end;
      if DataSize > 0 then
```

```
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFBMAC:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^ xor PByte(FBuffer)^;
        Inc(S);
        Dec(DataSize);
      end;
  end;
end;

procedure TCipher.DecodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F,B: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Decode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          XORBuffers(D, F, FBufSize, B);
          Decode(D);
          XORBuffers(D, F, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(FBuffer, D, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
```

```
      end;
    cmCBC:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          Move(D^, B^, FBufSize);
          Decode(D);
          XORBuffers(F, D, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(D, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^;
        D^ := S^ xor PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmOFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        D^ := S^ xor PByte(FBuffer)^;
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmCTSMAC, cmCBCMAC, cmCFBMAC:
      begin
        EncodeBuffer(Source, Dest, DataSize);
        Exit;
      end;
  end;
end;

procedure TCipher.CodeInit(Action: TPAction);
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
{  if (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) <> (Action = paCalc) then
    RaiseCipherException(errCantCalc, Format(sCantCalc, [ClassName]));}
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
  inherited CodeInit(Action);
```

```
end;

procedure TCipher.CodeDone(Action: TPAction);
begin
  inherited CodeDone(Action);
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
end;

procedure TCipher.CodeBuf(var Buffer; const BufferSize: Integer; Action:
TPAction);
begin
  if Action = paDecode then
  begin
    if Action in Actions then
      DecodeBuffer(Buffer, Buffer, BufferSize);
    inherited CodeBuf(Buffer, BufferSize, Action);
  end else
  begin
    inherited CodeBuf(Buffer, BufferSize, Action);
    if Action in Actions then
      EncodeBuffer(Buffer, Buffer, BufferSize);
  end;
end;

function TCipher.CalcMAC(Format: Integer): String;
var
  B: PByteArray;
begin
  if Mode in [cmECB, cmOFB] then
    RaiseCipherException(errInvalidMACMode, sInvalidMACMode);
  Done;
  B := AllocMem(FBufSize);
  try
    Move(FBuffer^, B^, FBufSize);
    EncodeBuffer(B^, B^, FBufSize);
    SetLength(Result, FBufSize);
    Move(FFeedback^, PChar(Result)^, FBufSize);
    if Protection <> nil then Result := Protection.CodeString(Result,
paScramble, Format)
      else Result := StrToFormat(PChar(Result), Length(Result), Format);
  finally
    ReallocMem(B, 0);
    Done;
  end;
end;

class procedure TCipher_Gost.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 32;
  AUserSize := 32;
end;

class function TCipher_Gost.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B3h,003h,0A0h,03Fh,0B5h,07Bh,091h,04Dh
         DB    097h,051h,024h,040h,0BDh,0CFh,025h,015h
         DB    034h,005h,09Ch,0F8h,0ABh,010h,086h,09Fh
         DB    0F2h,080h,047h,084h,047h,09Bh,01Ah,0D1h
end;

type
  PCipherRec = ^TCipherRec;
  TCipherRec = packed record
```

```
                  case Integer of
                    0: (X: array[0..7] of Byte);
                    1: (A, B: LongWord);
                end;

procedure TCipher_Gost.Encode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  K := User;
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := User;
    T := A + K[0];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[1];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Inc(PInteger(K), 2);
  end;
  K := @PIntArray(User)[6];
  for I := 0 to 3 do
  begin
    T := A + K[1];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[0];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Decode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  K := User;
  for I := 0 to 3 do
  begin
    T := A + K[0];
    B := B xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    T := B + K[1];
    A := A xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    Inc(PInteger(K), 2);
  end;
```

```
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := @PIntArray(User)[6];
    T := A + K[1];
    B := B xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    T := B + K[0];
    A := A xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  InitEnd(IVector);
end;

class procedure TCipher_Blowfish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 56;
  AUserSize := SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key);
end;

class function TCipher_Blowfish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    019h,071h,0CAh,0CDh,02Bh,09Ch,085h,029h
         DB    0DAh,081h,047h,0B7h,0EBh,0CEh,016h,0C6h
         DB    091h,00Eh,01Dh,0C8h,040h,012h,03Eh,035h
         DB    070h,0EDh,0BCh,096h,04Ch,013h,0D0h,0B8h
end;

type
  PBlowfish = ^TBlowfish;
  TBlowfish = array[0..3, 0..255] of LongWord;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
procedure TCipher_Blowfish.Encode386(Data: Pointer);
asm  // specaly for CPU < 486
         PUSH   EDI
         PUSH   ESI
         PUSH   EBX
         PUSH   EBP
         PUSH   EDX

         MOV    ESI,[EAX].TCipher_Blowfish.FUser

         MOV    EBX,[EDX]         // A
         MOV    EDX,[EDX + 4]     // B

         XCHG   BL,BH       // here BSWAP EBX,EDX
         XCHG   DL,DH
         ROL    EBX,16
         ROL    EDX,16
         XCHG   BL,BH
         XCHG   DL,DH
```

```
        XOR     EBX,[ESI + 4 * 256 * 4]
        XOR     EDI,EDI

@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP
        INC     EDI
        TEST    EDI,010h
        JZ      @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4 + 17 * 4]

        XCHG    BL,BH         // here BSWAP EBX,EDX
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;

procedure TCipher_Blowfish.Decode386(Data: Pointer);
asm // specaly for CPU < 486
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser

        MOV     EBX,[EDX]          // A
        MOV     EDX,[EDX + 4]      // B

        XCHG    BL,BH
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        XOR     EBX,[ESI + 4 * 256 * 4 + 17 * 4]

        MOV     EDI,16

@@1:    MOV     EAX,EBX
```

```
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP

        DEC     EDI
        JNZ     @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4]

        XCHG    BL,BH         // BSWAP
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;
  {$ENDIF} //486GE
{$ENDIF}

procedure TCipher_Blowfish.Encode(Data: Pointer);
{$IFDEF UseASM}  // спеціально для CPU >= 486
asm
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser
        MOV     EBX,[EDX]          // A
        MOV     EBP,[EDX + 4]      // B

        BSWAP   EBX                // CPU >= 486
        BSWAP   EBP

        XOR     EDI,EDI
        XOR     EBX,[ESI + 4 * 256 * 4]
//      XOR     ECX,ECX
@@1:

        MOV     EAX,EBX
        SHR     EBX,16
        MOVZX   ECX,BH     // it's faster with AMD Chips,
//        MOV     CL,BH    // it's faster with PII's
        MOV     EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
//        MOV     CL,BL
```

```
        ADD     EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
//        MOV     CL,AH
        XOR     EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
//        MOV     CL,AL
        ADD     EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR     EBP,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        INC     EDI
        XOR     EDX,EBP
        TEST    EDI,010h
        MOV     EBP,EAX
        MOV     EBX,EDX
        JZ      @@1

        POP     EAX
        XOR     EBP,[ESI + 4 * 256 * 4 + 17 * 4]

        BSWAP   EBX
        BSWAP   EBP

        MOV     [EAX],EBP
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  A := SwapInteger(PCipherRec(Data).A) xor P[0]; Inc(PInteger(P));
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    B := B xor P[0] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
                         D[3, A        and $FF]);

    A := A xor P[1] xor (D[0, B shr 24        ] +
                         D[1, B shr 16 and $FF] xor
                         D[2, B shr  8 and $FF] +
                         D[3, B        and $FF]);
    Inc(PInteger(P), 2);
  end;
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser
```

```
        MOV     EBX,[EDX]          // A
        MOV     EBP,[EDX + 4]      // B

        BSWAP   EBX
        BSWAP   EBP

        XOR     EBX,[ESI + 4 * 256 * 4 + 17 * 4]
        MOV     EDI,16
//        XOR     ECX,ECX

@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
//        MOV     CL,BH
        MOV     EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
//        MOV     CL,BL
        ADD     EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
//        MOV     CL,AH
        XOR     EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
//        MOV     CL,AL
        ADD     EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR     EBP,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR     EDX,EBP
        DEC     EDI
        MOV     EBP,EAX
        MOV     EBX,EDX
        JNZ     @@1

        POP     EAX
        XOR     EBP,[ESI + 4 * 256 * 4]

        BSWAP   EBX
        BSWAP   EBP

        MOV     [EAX],EBP
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key) -
SizeOf(Integer));
  A := SwapInteger(PCipherRec(Data).A) xor P[0];
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    Dec(PInteger(P), 2);
    B := B xor P[1] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
                         D[3, A        and $FF]);
    A := A xor P[0] xor (D[0, B shr 24        ] +
                         D[1, B shr 16 and $FF] xor
                         D[2, B shr  8 and $FF] +
```

```
                        D[3, B        and $FF]);
  end;
  Dec(PInteger(P));
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Init(const Key; Size: Integer; IVector: Pointer);
var
  I,J: Integer;
  B: array[0..7] of Byte;
  K: PByteArray;
  P: PIntArray;
  S: PBlowfish;
begin
  InitBegin(Size);
  K := @Key;
  S := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  Move(Blowfish_Data, S^, SizeOf(Blowfish_Data));
  Move(Blowfish_Key, P^, Sizeof(Blowfish_Key));
  J := 0;
  for I := 0 to 17 do
  begin
    P[I] := P[I] xor (K[(J + 0) mod Size] shl 24 +
                      K[(J + 1) mod Size] shl 16 +
                      K[(J + 2) mod Size] shl  8 +
                      K[(J + 3) mod Size]);
    J := (J + 4) mod Size;
  end;
  FillChar(B, SizeOf(B), 0);
  for I := 0 to 8 do
  begin
    Encode(@B);
    P[I * 2]     := SwapInteger(PCipherRec(@B).A);
    P[I * 2 + 1] := SwapInteger(PCipherRec(@B).B);
  end;
  for I := 0 to 3 do
    for J := 0 to 127 do
    begin
      Encode(@B);
      S[I, J * 2]    := SwapInteger(PCipherRec(@B).A);
      S[I, J * 2 +1] := SwapInteger(PCipherRec(@B).B);
    end;

  FillChar(B, SizeOf(B), 0);
  InitEnd(IVector);
end;

class procedure TCipher_IDEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 208;
end;

class function TCipher_IDEA.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    08Ch,065h,0CAh,0D8h,043h,0E7h,099h,093h
        DB    0EDh,041h,0EAh,048h,0FDh,066h,050h,094h
        DB    0A2h,025h,06Dh,0D7h,0B1h,0D0h,09Ah,023h
        DB    03Dh,0D2h,0E8h,0ECh,0C9h,045h,07Fh,07Eh
end;

function IDEAMul(X, Y: LongWord): LongWord; assembler; register;
```

```
asm
     AND    EAX,0FFFFh
     JZ     @@1
     AND    EDX,0FFFFh
     JZ     @@1
     MUL    EDX
     MOV    ECX,EAX
     MOV    EDX,EAX
     SHR    EDX,16
     SUB    EAX,EDX
     CMP    AX,CX
     JNA    @@2
     INC    EAX
@@2: RET
@@1: MOV    ECX,1
     SUB    ECX,EAX
     SUB    ECX,EDX
     MOV    EAX,ECX
end;

procedure TCipher_IDEA.Cipher(Data, Key: PWordArray);
var
  I: LongWord;
  X,Y,A,B,C,D: LongWord;
begin
  I := SwapInteger(PIntArray(Data)[0]);
  A := LongRec(I).Hi;
  B := LongRec(I).Lo;
  I := SwapInteger(PIntArray(Data)[1]);
  C := LongRec(I).Hi;
  D := LongRec(I).Lo;
  for I := 0 to 7 do
  begin
    A := IDEAMul(A, Key[0]);
    Inc(B, Key[1]);
    Inc(C, Key[2]);
    D := IDEAMul(D, Key[3]);
    Y := C xor A;
    Y := IDEAMul(Y, Key[4]);
    X := B xor D + Y;
    X := IDEAMul(X, Key[5]);
    Inc(Y, X);
    A := A xor X;
    D := D xor Y;
    Y := B xor Y;
    B := C xor X;
    C := Y;
    Inc(PWord(Key), 6);
  end;
  LongRec(I).Hi := IDEAMul(A, Key[0]);
  LongRec(I).Lo := C + Key[1];
  PIntArray(Data)[0] := SwapInteger(I);
  LongRec(I).Hi := B + Key[2];
  LongRec(I).Lo := IDEAMul(D, Key[3]);
  PIntArray(Data)[1] := SwapInteger(I);
end;

procedure TCipher_IDEA.Encode(Data: Pointer);
begin
  Cipher(Data, User);
end;

procedure TCipher_IDEA.Decode(Data: Pointer);
begin
  Cipher(Data, @PIntArray(User)[26]);
end;

procedure TCipher_IDEA.Init(const Key; Size: Integer; IVector: Pointer);
```

```
  function IDEAInv(X: Word): Word;
  var
    A, B, C, D: Word;
  begin
    if X <= 1 then
    begin
      Result := X;
      Exit;
    end;
    A := 1;
    B := $10001 div X;
    C := $10001 mod X;
    while C <> 1 do
    begin
      D := X div C;
      X := X mod C;
      Inc(A, B * D);
      if X = 1 then
      begin
        Result := A;
        Exit;
      end;
      D := C div X;
      C := C mod X;
      Inc(B, A * D);
    end;
    Result := 1 - B;
  end;

var
  I: Integer;
  E: PWordArray;
  A,B,C: Word;
  K,D: PWordArray;
begin
  InitBegin(Size);
  E := User;
  Move(Key, E^, Size);
  for I := 0 to 7 do E[I] := Swap(E[I]);
  for I := 0 to 39 do
    E[I + 8] := E[I and not 7 + (I + 1) and 7] shl 9 or
                E[I and not 7 + (I + 2) and 7] shr 7;
  for I := 41 to 44 do
    E[I + 7] := E[I] shl 9 or E[I + 1] shr 7;
  K  := E;
  D  := @E[100];
  A  := IDEAInv(K[0]);
  B  := 0 - K[1];
  C  := 0 - K[2];
  D[3] := IDEAInv(K[3]);
  D[2] := C;
  D[1] := B;
  D[0] := A;
  Inc(PWord(K), 4);
  for I := 1 to 8 do
  begin
    Dec(PWord(D), 6);
    A    := K[0];
    D[5] := K[1];
    D[4] := A;
    A    := IDEAInv(K[2]);
    B    := 0 - K[3];
    C    := 0 - K[4];
    D[3] := IDEAInv(K[5]);
    D[2] := B;
    D[1] := C;
    D[0] := A;
    Inc(PWord(K), 6);
  end;
```

```
  A := D[2]; D[2] := D[1]; D[1] := A;
  InitEnd(IVector);
end;

type
  PSAFERRec = ^TSAFERRec;
  TSAFERRec = packed record
                case Integer of
                  0: (A,B,C,D,E,F,G,H: Byte);
                  1: (X,Y: Integer);
              end;

procedure TCipher_SAFER.SetRounds(Value: Integer);
begin
  if (Value < 4) or (Value > 13) then
    case FSaferMode of
      smK40, smSK40: Value := 5;
      smK64, smSK64: Value := 6;
      smK128, smSK128: Value := 10;
    else
      Value := 8;
    end;
  FRounds := Value;
end;

class procedure TCipher_SAFER.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 768;
end;

class function TCipher_SAFER.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    000h,03Dh,049h,020h,073h,063h,085h,0AAh
         DB    0D9h,0C2h,00Ah,0DEh,07Eh,09Eh,0E9h,0ABh
         DB    024h,0D0h,074h,034h,047h,07Eh,021h,01Dh
         DB    055h,0F9h,035h,028h,098h,084h,0A8h,075h
end;

procedure TCipher_SAFER.Encode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 512);
  with PSAFERRec(Data)^ do
  begin
    for I := 1 to FRounds do
    begin
      A := A xor Key[0];
      B := B  +  Key[1];
      C := C  +  Key[2];
      D := D xor Key[3];
      E := E xor Key[4];
      F := F  +  Key[5];
      G := G  +  Key[6];
      H := H xor Key[7];

      A := Exp[A]  +  Key[8];
      B := Log[B] xor Key[9];
      C := Log[C] xor Key[10];
      D := Exp[D]  +  Key[11];
```

```
      E := Exp[E]  +  Key[12];
      F := Log[F] xor Key[13];
      G := Log[G] xor Key[14];
      H := Exp[H]  +  Key[15];

      Inc(B, A); Inc(A, B);
      Inc(D, C); Inc(C, D);
      Inc(F, E); Inc(E, F);
      Inc(H, G); Inc(G, H);

      Inc(C, A); Inc(A, C);
      Inc(G, E); Inc(E, G);
      Inc(D, B); Inc(B, D);
      Inc(H, F); Inc(F, H);

      Inc(E, A); Inc(A, E);
      Inc(F, B); Inc(B, F);
      Inc(G, C); Inc(C, G);
      Inc(H, D); Inc(D, H);

      T := B; B := E; E := C; C := T;
      T := D; D := F; F := G; G := T;

      Inc(PByte(Key), 16);
    end;
    A := A xor Key[0];
    B := B  +  Key[1];
    C := C  +  Key[2];
    D := D xor Key[3];
    E := E xor Key[4];
    F := F  +  Key[5];
    G := G  +  Key[6];
    H := H xor Key[7];
  end;
end;

procedure TCipher_SAFER.Decode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 504 + 8 * (FRounds * 2 + 1));
  with PSAFERRec(Data)^ do
  begin
    H := H xor Key[7];
    G := G  -  Key[6];
    F := F  -  Key[5];
    E := E xor Key[4];
    D := D xor Key[3];
    C := C  -  Key[2];
    B := B  -  Key[1];
    A := A xor Key[0];

    for I := 1 to FRounds do
    begin
      Dec(PByte(Key), 16);
      T := E; E := B; B := C; C := T;
      T := F; F := D; D := G; G := T;

      Dec(A, E); Dec(E, A);
      Dec(B, F); Dec(F, B);
      Dec(C, G); Dec(G, C);
      Dec(D, H); Dec(H, D);

      Dec(A, C); Dec(C, A);
      Dec(E, G); Dec(G, E);
```

```
      Dec(B, D); Dec(D, B);
      Dec(F, H); Dec(H, F);

      Dec(A, B); Dec(B, A);
      Dec(C, D); Dec(D, C);
      Dec(E, F); Dec(F, E);
      Dec(G, H); Dec(H, G);

      H := H  -  Key[15];
      G := G xor Key[14];
      F := F xor Key[13];
      E := E  -  Key[12];
      D := D  -  Key[11];
      C := C xor Key[10];
      B := B xor Key[9];
      A := A  -  Key[8];

      H := Log[H] xor Key[7];
      G := Exp[G]  -  Key[6];
      F := Exp[F]  -  Key[5];
      E := Log[E] xor Key[4];
      D := Log[D] xor Key[3];
      C := Exp[C]  -  Key[2];
      B := Exp[B]  -  Key[1];
      A := Log[A] xor Key[0];
    end;
  end;
end;

procedure TCipher_SAFER.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smStrong);
end;

procedure TCipher_SAFER.InitNew(const Key; Size: Integer; IVector: Pointer;
SAFERMode: TSAFERMode);

  procedure InitTab;
  var
    I,E: Integer;
    Exp: PByte;
    Log: PByteArray;
  begin
    Exp := User;
    Log := Pointer(PChar(User) + 256);
    E   := 1;
    for I := 0 to 255 do
    begin
      Exp^ := E and $FF;
      Log[E and $FF] := I;
      E := (E * 45) mod 257;
      Inc(Exp);
    end;
  end;

  procedure InitKey;

    function ROR3(Value: Byte): Byte; assembler;
    asm
      ROR AL,3
    end;

    function ROL6(Value: Byte): Byte; assembler;
    asm
      ROL AL,6
    end;

  var
    D: PByte;
```

```
    Exp: PByteArray;
    Strong: Boolean;
    K: array[Boolean, 0..8] of Byte;
    I,J: Integer;
  begin
    Strong := FSAFERMode in [smStrong, smSK40, smSK64, smSK128];
    Exp := User;
    D := User;
    Inc(D, 512);
    FillChar(K, SizeOf(K), 0);
{Встановлюється ключ A}
    I := Size;
    if I > 8 then I := 8;
    Move(Key, K[False], I);

    if FSAFERMode in [smK40, smSK40] then
    begin
      K[False, 5] := K[False, 0] xor K[False, 2] xor 129;
      K[False, 6] := K[False, 0] xor K[False, 3] xor K[False, 4] xor 66;
      K[False, 7] := K[False, 1] xor K[False, 2] xor K[False, 4] xor 36;
      K[False, 8] := K[False, 1] xor K[False, 3] xor 24;
      Move(K[False], K[True], SizeOf(K[False]));
    end else
    begin
      if Size > 8 then
      begin
        I := Size - 8;
        if I > 8 then I := 8;
        Move(TByteArray(Key)[8], K[True], I);
      end else Move(K[False], K[True], 9);
      for I := 0 to 7 do
      begin
        K[False, 8] := K[False, 8] xor K[False, I];
        K[True, 8]  := K[True, 8]  xor K[True, I];
      end;
    end;
{Встановлюються дані ключа}
    Move(K[True], D^, 8);
    Inc(D, 8);

    for I := 0 to 8 do K[False, I] := ROR3(K[False, I]);

    for I := 1 to FRounds do
    begin
      for J := 0 to 8 do
      begin
        K[False, J] := ROL6(K[False, J]);
        K[True, J] := ROL6(K[True, J]);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[False, (J + I * 2 -1) mod 9] + Exp[Exp[18 * I + J
+1]]
          else D^ := K[False, J] + Exp[Exp[18 * I + J +1]];
        Inc(D);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[True, (J + I * 2) mod 9] + Exp[Exp[18 * I + J
+10]]
          else D^ := K[True, J] + Exp[Exp[18 * I + J +10]];
        Inc(D);
      end;
    end;
    FillChar(K, SizeOf(K), 0);
  end;

begin
  InitBegin(Size);
```

```
  FSAFERMode := SAFERMode;
  if SAFERMode = smDefault then
    if Size <= 5 then FSAFERMode := smK40 else
      if Size <= 8 then FSAFERMode := smK64 else FSAFERMode := smK128
  else
    if SAFERMode = smStrong then
      if Size <= 5 then FSAFERMode := smSK40 else
        if Size <= 8 then FSAFERMode := smSK64 else FSAFERMode := smSK128;
  SetRounds(FRounds);
  InitTab;
  InitKey;
  InitEnd(IVector);
end;

class procedure TCipher_SAFER_K40.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 5;
end;

class function TCipher_SAFER_K40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    005h,0B4h,019h,057h,026h,05Ch,013h,060h
         DB    0A0h,082h,094h,045h,0D6h,0A5h,046h,0D8h
         DB    073h,050h,096h,080h,04Fh,06Dh,0F7h,0E5h
         DB    0C8h,01Ah,0EFh,044h,04Ch,0B4h,059h,013h
end;

procedure TCipher_SAFER_K40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK40);
end;

class function TCipher_SAFER_SK40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0D9h,003h,003h,06Dh,018h,038h,0D1h,0C1h
         DB    089h,0E8h,038h,012h,07Fh,028h,0FCh,0C7h
         DB    0C5h,00Bh,0B7h,0C4h,0DBh,021h,0A4h,031h
         DB    020h,008h,08Ah,077h,0F7h,0DFh,026h,0FFh
end;

procedure TCipher_SAFER_SK40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK40);
end;

class procedure TCipher_SAFER_K64.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 8;
end;

class function TCipher_SAFER_K64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,0B2h,032h,0F0h,00Eh,0C2h,0DAh,0CBh
         DB    039h,008h,02Dh,05Ch,093h,0FFh,0CEh,0F3h
         DB    08Fh,01Fh,0B7h,02Ch,0C5h,0C7h,0A7h,0E9h
         DB    089h,0BEh,061h,08Bh,000h,0E6h,09Fh,00Eh
end;

procedure TCipher_SAFER_K64.Init(const Key; Size: Integer; IVector: Pointer);
```

```
begin
  InitNew(Key, Size, IVector, smK64);
end;

class function TCipher_SAFER_SK64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0DDh,09Ch,01Ah,0D6h,029h,00Ch,0EEh,04Fh
         DB    0E5h,04Bh,0C0h,055h,0BFh,022h,00Eh,0BCh
         DB    019h,041h,078h,0CFh,094h,0DBh,02Fh,039h
         DB    06Bh,01Eh,0A7h,0CAh,04Bh,05Fh,077h,0E0h
end;

procedure TCipher_SAFER_SK64.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK64);
end;

class procedure TCipher_SAFER_K128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 16;
end;

class function TCipher_SAFER_K128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    00Ch,0A9h,070h,0B9h,0F3h,014h,087h,0D9h
         DB    09Eh,05Eh,078h,031h,074h,0DFh,0A8h,0BBh
         DB    03Dh,040h,0A5h,0D9h,08Ch,07Ch,004h,0B7h
         DB    09Ch,001h,0DAh,063h,0ABh,026h,035h,0BCh
end;

procedure TCipher_SAFER_K128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK128);
end;

class function TCipher_SAFER_SK128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0C8h,0A6h,070h,033h,029h,038h,038h,02Bh
         DB    069h,0ACh,061h,072h,08Fh,0DCh,09Fh,0A4h
         DB    09Eh,06Fh,0C4h,053h,0D8h,089h,0FFh,042h
         DB    072h,009h,07Dh,0CDh,0D0h,0EAh,07Eh,028h
end;

procedure TCipher_SAFER_SK128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK128);
end;

type
  PTEARec = ^TTEARec;
  TTEARec = packed record
              A,B,C,D: LongWord;
            end;

const
  TEA_Delta = $9E3779B9;

procedure TCipher_TEA.SetRounds(Value: Integer);
begin
  FRounds := Value;
  if FRounds < 16 then FRounds := 16 else
```

```
    if FRounds > 32 then FRounds := 32;
end;

class procedure TCipher_TEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 32;
end;

class function TCipher_TEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B7h,0B8h,0AAh,0BBh,026h,04Bh,006h,0F9h
         DB    070h,086h,0B0h,0E4h,056h,004h,029h,0CCh
         DB    0BFh,055h,0EAh,04Eh,0EFh,059h,026h,018h
         DB    019h,0B0h,003h,07Ch,029h,08Ch,0E2h,077h
end;

procedure TCipher_TEA.Encode(Data: Pointer);
{$IFDEF UseASM}
asm
         PUSH  EDI
         PUSH  ESI
         PUSH  EBX
         PUSH  EBP
         PUSH  EDX

         MOV   EBX,[EDX]       // X
         MOV   EDX,[EDX + 4]   // Y
         XOR   EDI,EDI         // Sum

         MOV   ESI,[EAX].TCipher_TEA.FUser  // User
         MOV   ECX,[EAX].TCipher_TEA.FRounds  // Rounds

@@1:     ADD   EDI,TEA_Delta

         MOV   EAX,EDX
         MOV   EBP,EDX
         SHL   EAX,4
         SHR   EBP,5
         ADD   EAX,[ESI]
         ADD   EBP,[ESI +  4]
         XOR   EAX,EDX
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EAX,EBX
         MOV   EBX,EAX
         SHL   EAX,4
         MOV   EBP,EBX
         SHR   EBP,5
         ADD   EAX,[ESI +  8]
         XOR   EAX,EBX
         ADD   EBP,[ESI + 12]
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EDX,EAX

         DEC   ECX
         JNZ   @@1

         POP   EAX
         MOV   [EAX],EBX
         MOV   [EAX + 4],EDX
```

```
          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Inc(Sum, TEA_Delta);
      Inc(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Inc(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
          PUSH  EDI
          PUSH  ESI
          PUSH  EBX
          PUSH  EBP
          PUSH  EDX

          MOV   EBX,[EDX]       // X
          MOV   EDX,[EDX + 4]   // Y

          MOV   ESI,[EAX].TCipher_TEA.FUser  // User
          MOV   EDI,TEA_Delta
          MOV   ECX,[EAX].TCipher_TEA.FRounds  // Rounds
          IMUL  EDI,ECX

@@1:      MOV   EAX,EBX
          MOV   EBP,EBX
          SHL   EAX,4
          SHR   EBP,5
          ADD   EAX,[ESI +  8]
          ADD   EBP,[ESI + 12]
          XOR   EAX,EBX
          ADD   EAX,EDI
          XOR   EAX,EBP
          SUB   EDX,EAX
          MOV   EAX,EDX
          SHL   EAX,4
          MOV   EBP,EDX
          SHR   EBP,5
          ADD   EAX,[ESI]
          XOR   EAX,EDX
          ADD   EBP,[ESI + 4]
          ADD   EAX,EDI

          XOR   EAX,EBP
          SUB   EDI,TEA_Delta
          SUB   EBX,EAX

          DEC   ECX
          JNZ   @@1

          POP   EAX
          MOV   [EAX],EBX
```

```
          MOV   [EAX + 4],EDX

          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
      Dec(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Dec(Sum, TEA_Delta);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  SetRounds(FRounds);
  InitEnd(IVector);
end;

class function TCipher_TEAN.TestVector: Pointer;
asm
          MOV   EAX,OFFSET @Vector
          RET
@Vector:  DB    0CDh,07Eh,0BBh,0A2h,092h,01Ah,04Bh,03Bh
          DB    0E2h,09Eh,062h,0CFh,0F7h,01Dh,0A5h,0DFh
          DB    063h,033h,094h,029h,0E2h,036h,07Ch,066h
          DB    03Fh,0F8h,01Ah,0F9h,002h,078h,0BFh,0A1h
end;

procedure TCipher_TEAN.Encode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  for I := 1 to FRounds do
  begin
    Inc(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    Inc(Sum, TEA_Delta);
    Inc(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
  end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

procedure TCipher_TEAN.Decode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := TEA_Delta * LongWord(FRounds);
```

```
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
      Dec(Sum, TEA_Delta);
      Dec(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

const
  SCOP_SIZE = 32; {це максімум}

class procedure TCipher_SCOP.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := SCOP_SIZE * SizeOf(Integer);
  AKeySize := 48;
  AUserSize := (384 * 4 + 4 * SizeOf(Integer)) * 2;
end;

class function TCipher_SCOP.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    014h,0C0h,009h,0E8h,073h,0B6h,053h,092h
         DB    08Bh,013h,069h,0A9h,0F2h,099h,0FEh,05Eh
         DB    0EEh,03Bh,0FDh,0C1h,050h,059h,00Eh,094h
         DB    062h,017h,008h,01Eh,0A4h,01Ah,04Dh,08Fh
end;

procedure TCipher_SCOP.Encode(Data: Pointer);
var
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
  B: PInteger;
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Inc(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Decode(Data: Pointer);
var
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
  B: PInteger;
```

```
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Dec(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Init(const Key; Size: Integer; IVector: Pointer);
var
  Init_State: packed record
                Coef: array[0..7, 0..3] of Byte;
                X: array[0..3] of LongWord;
              end;

  procedure ExpandKey;
  var
    P: PByteArray;
    I,C: Integer;
  begin
    C := 1;
    P := @Init_State;
    Move(Key, P^, Size);
    for I := Size to 47 do P[I] := P[I - Size] + P[I - Size +1];
    for I := 0 to 31 do
      if P[I] = 0 then
      begin
        P[I] := C;
        Inc(C);
      end;
  end;

  procedure GP8(Data: PIntArray);
  var
    I,I2: Integer;
    NewX: array[0..3] of LongWord;
    X1,X2,X3,X4: LongWord;
    Y1,Y2: LongWord;
  begin
    I := 0;
    while I < 8 do
    begin
      I2 := I shr 1;
      X1 := Init_State.X[I2] shr 16;
      X2 := X1 * X1;
      X3 := X2 * X1;
      X4 := X3 * X1;
      Y1 := Init_State.Coef[I][0] * X4 +
            Init_State.Coef[I][1] * X3 +
            Init_State.Coef[I][2] * X2 +
            Init_State.Coef[I][3] * X1 + 1;
      X1 := Init_State.X[I2] and $FFFF;
      X2 := X1 * X1;
      X3 := X2 * X1;
      X4 := X3 * X1;
```

```
      Y2 := Init_State.Coef[I +1][0] * X4 +
            Init_State.Coef[I +2][1] * X3 +
            Init_State.Coef[I +3][2] * X2 +
            Init_State.Coef[I +4][3] * X1 + 1;
      Data[I2] := Y1 shl 16 or Y2 and $FFFF;
      NewX[I2] := Y1 and $FFFF0000 or Y2 shr 16;
      Inc(I, 2);
    end;
    Init_State.X[0] := NewX[0] shr 16 or NewX[3] shl 16;
    Init_State.X[1] := NewX[0] shl 16 or NewX[1] shr 16;
    Init_State.X[2] := NewX[1] shl 16 or NewX[2] shr 16;
    Init_State.X[3] := NewX[2] shl 16 or NewX[3] shr 16;
  end;

var
  I,J: Integer;
  T: array[0..3] of Integer;
  P: PIntArray;
begin
  InitBegin(Size);
  FillChar(Init_State, SizeOf(Init_State), 0);
  FillChar(T, SizeOf(T), 0);
  P := Pointer(PChar(User) + 12);
  ExpandKey;
  for I := 0 to 7 do GP8(@T);
  for I := 0 to 11 do
  begin
    for J := 0 to 7 do GP8(@P[I * 32 + J * 4]);
    GP8(@T);
  end;
  GP8(@T);
  I := T[3] and $7F;
  P[I] := P[I] or 1;
  P := User;
  P[0] := T[3] shr 24;
  P[1] := T[3] shr 16;
  P[2] := T[3] shr  8;
  FillChar(Init_State, SizeOf(Init_State), 0);
  InitEnd(IVector);
  P := Pointer(PChar(User) + FUserSize shr 1);
  Move(User^, P^, FUserSize shr 1);
end;

procedure TCipher_SCOP.Done;
begin
  inherited Done;
  Move(PByteArray(User)[FUserSize shr 1], User^, FUserSize shr 1);
end;

class procedure TCipher_Q128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 256;
end;

class function TCipher_Q128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    099h,0AAh,0D0h,03Dh,0CAh,014h,04Eh,02Ah
         DB    0F8h,01Eh,001h,0A0h,0EAh,0ABh,09Fh,048h
         DB    023h,02Dh,059h,054h,054h,07Eh,02Bh,012h
         DB    086h,080h,0E8h,033h,0EBh,0E1h,05Eh,0AEh
end;

procedure TCipher_Q128.Encode(Data: Pointer);
{$IFDEF UseASM}
```

```
asm
        PUSH    ESI
        PUSH    EDI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     EDI,[EAX].TCipher_Q128.FUser

        MOV     EAX,[EDX]       // B0
        MOV     EBX,[EDX +  4]  // B1
        MOV     ECX,[EDX +  8]  // B2
        MOV     EDX,[EDX + 12]  // B3

        MOV     EBP,16

@@1:    MOV     ESI,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     ESI,10
        ADD     EAX,[EDI]
        XOR     EAX,EBX

        MOV     EBX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     EBX,10
        ADD     EAX,[EDI + 4]
        XOR     EAX,ECX

        MOV     ECX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     ECX,10
        ADD     EAX,[EDI + 8]
        XOR     EAX,EDX

        MOV     EDX,EAX
        AND     EAX,03FFh
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ROL     EDX,10
        ADD     EAX,[EDI + 12]
        XOR     EAX,ESI

        ADD     EDI,16

        DEC     EBP
        JNZ     @@1

        POP     ESI

        MOV     [ESI],EAX       // B0
        MOV     [ESI +  4],EBX  // B1
        MOV     [ESI +  8],ECX  // B2
        MOV     [ESI + 12],EDX  // B3


        POP     EBP
        POP     EBX
        POP     EDI
        POP     ESI
end;
{$ELSE}
var
  D: PInteger;
  B0, B1, B2, B3, I: LongWord;
begin
  D  := User;
  B0 := PIntArray(Data)[0];
```

```
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B1 := B1 xor (Q128_Data[B0 and $03FF] + D^); Inc(D); B0 := B0 shl 10 or B0
shr 22;
    B2 := B2 xor (Q128_Data[B1 and $03FF] + D^); Inc(D); B1 := B1 shl 10 or B1
shr 22;
    B3 := B3 xor (Q128_Data[B2 and $03FF] + D^); Inc(D); B2 := B2 shl 10 or B2
shr 22;
    B0 := B0 xor (Q128_Data[B3 and $03FF] + D^); Inc(D); B3 := B3 shl 10 or B3
shr 22;
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}
procedure TCipher_Q128.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
       PUSH   ESI
       PUSH   EDI
       PUSH   EBX
       PUSH   EBP
       PUSH   EDX

       MOV    EDI,[EAX].TCipher_Q128.FUser
       LEA    EDI,[EDI + 64 * 4]

       MOV    ESI,[EDX]       // B0
       MOV    EBX,[EDX +  4]  // B1
       MOV    ECX,[EDX +  8]  // B2
       MOV    EDX,[EDX + 12]  // B3

       MOV    EBP,16

@@1:   SUB    EDI,16

       ROR    EDX,10
       MOV    EAX,EDX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI + 12]
       XOR    ESI,EAX

       ROR    ECX,10
       MOV    EAX,ECX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  8]
       XOR    EDX,EAX

       ROR    EBX,10
       MOV    EAX,EBX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  4]
       XOR    ECX,EAX

       ROR    ESI,10
       MOV    EAX,ESI
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI]
       XOR    EBX,EAX
```

```
        DEC     EBP
        JNZ     @@1

        POP     EAX

        MOV     [EAX],ESI       // B0
        MOV     [EAX +  4],EBX  // B1
        MOV     [EAX +  8],ECX  // B2
        MOV     [EAX + 12],EDX  // B3


        POP     EBP
        POP     EBX
        POP     EDI
        POP     ESI
end;
{$ELSE}
var
  D: PInteger;
  B0, B1, B2, B3, I: LongWord;
begin
  D  := @PIntArray(User)[63];
  B0 := PIntArray(Data)[0];
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B3 := B3 shr 10 or B3 shl 22; B0 := B0 xor (Q128_Data[B3 and $03FF] + D^);
Dec(D);
    B2 := B2 shr 10 or B2 shl 22; B3 := B3 xor (Q128_Data[B2 and $03FF] + D^);
Dec(D);
    B1 := B1 shr 10 or B1 shl 22; B2 := B2 xor (Q128_Data[B1 and $03FF] + D^);
Dec(D);
    B0 := B0 shr 10 or B0 shl 22; B1 := B1 xor (Q128_Data[B0 and $03FF] + D^);
Dec(D);
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}

procedure TCipher_Q128.Init(const Key; Size: Integer; IVector: Pointer);
var
  K: array[0..3] of LongWord;
  I: Integer;
  D: PInteger;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  D := User;
  for I := 19 downto 1 do
  begin
    K[1] := K[1] xor Q128_Data[K[0] and $03FF]; K[0] := K[0] shr 10 or K[0] shl
22;
    K[2] := K[2] xor Q128_Data[K[1] and $03FF]; K[1] := K[1] shr 10 or K[1] shl
22;
    K[3] := K[3] xor Q128_Data[K[2] and $03FF]; K[2] := K[2] shr 10 or K[2] shl
22;
    K[0] := K[0] xor Q128_Data[K[3] and $03FF]; K[3] := K[3] shr 10 or K[3] shl
22;
    if I <= 16 then
    begin
      D^ := K[0]; Inc(D);
      D^ := K[1]; Inc(D);
      D^ := K[2]; Inc(D);
```

```
      D^ := K[3]; Inc(D);
    end;
  end;
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

type
  P3Way_Key = ^T3Way_Key;
  T3Way_Key = packed record
                E_Key: array[0..2] of Integer;
                E_Data: array[0..11] of Integer;
                D_Key: array[0..2] of Integer;
                D_Data: array[0..11] of Integer;
              end;

class procedure TCipher_3Way.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 12;
  AKeySize := 12;
  AUserSize := SizeOf(T3Way_Key);
end;

class function TCipher_3Way.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    077h,0FCh,077h,094h,07Ch,08Fh,0DEh,021h
         DB    0E9h,081h,0DFh,02Ah,0B1h,0BCh,07Eh,0F8h
         DB    0A3h,0B6h,044h,04Bh,0B6h,0FCh,079h,0C4h
         DB    09Bh,068h,04Fh,009h,0C7h,0BFh,00Eh,005h
end;

procedure TCipher_3Way.Encode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := E_Key[0];
    K1 := E_Key[1];
    K2 := E_Key[2];
    E  := @E_Data;
  end;
  A0 := PIntArray(Data)[0];
  A1 := PIntArray(Data)[1];
  A2 := PIntArray(Data)[2];
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    asm
```

```
      ROR  B0,10
      ROL  B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  PIntArray(Data)[0] := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl
16 xor
                              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl
24 xor
                              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl
8;
  PIntArray(Data)[1] := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl
16 xor
                              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl
24 xor
                              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl
8;
  PIntArray(Data)[2] := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl
16 xor
                              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl
24 xor
                              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl
8;
end;

procedure TCipher_3Way.Decode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := D_Key[0];
    K1 := D_Key[1];
    K2 := D_Key[2];
    E  := @D_Data;
  end;
  A0 := SwapBits(PIntArray(Data)[2]);
  A1 := SwapBits(PIntArray(Data)[1]);
  A2 := SwapBits(PIntArray(Data)[0]);
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    asm
```

```
      ROR B0,10
      ROL B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
  B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
  B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;

  PIntArray(Data)[2] := SwapBits(B0);
  PIntArray(Data)[1] := SwapBits(B1);
  PIntArray(Data)[0] := SwapBits(B2);
end;

procedure TCipher_3Way.Init(const Key; Size: Integer; IVector: Pointer);

  procedure RANDGenerate(Start: Integer; var P: Array of Integer);
  var
    I: Integer;
  begin
    for I := 0 to 11 do
    begin
      P[I] := Start;
      Start := Start shl 1;
      if Start and $10000 <> 0 then Start := Start xor $11011;
    end;
  end;

var
  A0, A1, A2: Integer;
  B0, B1, B2: Integer;
begin
  InitBegin(Size);
  with P3Way_Key(User)^ do
  begin
    Move(Key, E_Key, Size);
    Move(Key, D_Key, Size);
    RANDGenerate($0B0B, E_Data);
    RANDGenerate($B1B1, D_Data);

    A0 := D_Key[0]; A1 := D_Key[1]; A2 := D_Key[2];
    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    D_Key[2] := SwapBits(B0); D_Key[1] := SwapBits(B1); D_Key[0] :=
SwapBits(B2);
  end;
  InitEnd(IVector);
```

```
end;

class procedure TCipher_Twofish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 32;
  AUserSize := 4256;
end;

class function TCipher_Twofish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0A5h,053h,057h,003h,0EFh,033h,048h,079h
         DB    09Fh,022h,0B4h,054h,097h,005h,084h,019h
         DB    087h,0BDh,083h,01Ch,04Dh,0AEh,012h,013h
         DB    060h,07Ch,07Ch,0D1h,098h,045h,002h,019h
end;

type
  PTwofishBox = ^TTwofishBox;
  TTwofishBox = array[0..3, 0..255] of Longword;

  TLongRec = record
               case Integer of
                 0: (L: Longword);
                 1: (A,B,C,D: Byte);
             end;

procedure TCipher_Twofish.Encode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S   := User;
  A.L := PIntArray(Data)[0] xor S[0];
  B.L := PIntArray(Data)[1] xor S[1];
  C.L := PIntArray(Data)[2] xor S[2];
  D.L := PIntArray(Data)[3] xor S[3];

  S   := @PIntArray(User)[8];
  Box := @PIntArray(User)[40];
  for I := 0 to 7 do
  begin
    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C] xor Box[0, B.D];
    asm ROL  D.L,1 end;
    C.L := C.L xor (X + Y       + S[0]);
    D.L := D.L xor (X + Y shl 1 + S[1]);
    asm ROR  C.L,1 end;

    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C] xor Box[0, D.D];
    asm ROL  B.L,1 end;
    A.L := A.L xor (X + Y       + S[2]);
    B.L := B.L xor (X + Y shl 1 + S[3]);
    asm ROR  A.L,1 end;
    Inc(PInteger(S), 4);
  end;
  S := User;
  PIntArray(Data)[0] := C.L xor S[4];
  PIntArray(Data)[1] := D.L xor S[5];
  PIntArray(Data)[2] := A.L xor S[6];
  PIntArray(Data)[3] := B.L xor S[7];
end;
```

```
procedure TCipher_Twofish.Decode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S := User;
  Box := @PIntArray(User)[40];
  C.L := PIntArray(Data)[0] xor S[4];
  D.L := PIntArray(Data)[1] xor S[5];
  A.L := PIntArray(Data)[2] xor S[6];
  B.L := PIntArray(Data)[3] xor S[7];
  S := @PIntArray(User)[36];
  for I := 0 to 7 do
  begin
    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[0, D.D] xor Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C];
    asm ROL  A.L,1 end;
    B.L := B.L xor (X + Y shl 1 + S[3]);
    A.L := A.L xor (X + Y       + S[2]);
    asm ROR  B.L,1 end;

    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[0, B.D] xor Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C];
    asm ROL  C.L,1 end;
    D.L := D.L xor (X + Y shl 1 + S[1]);
    C.L := C.L xor (X + Y       + S[0]);
    asm ROR  D.L,1 end;
    Dec(PByte(S),16);
  end;
  S := User;
  PIntArray(Data)[0] := A.L xor S[0];
  PIntArray(Data)[1] := B.L xor S[1];
  PIntArray(Data)[2] := C.L xor S[2];
  PIntArray(Data)[3] := D.L xor S[3];
end;

procedure TCipher_Twofish.Init(const Key; Size: Integer; IVector: Pointer);
var
  BoxKey: array[0..3] of TLongRec;
  SubKey: PIntArray;
  Box: PTwofishBox;

  procedure SetupKey;

    function Encode(K0, K1: Integer): Integer;
    var
      R, I, J, G2, G3: Integer;
      B: byte;
    begin
      R := 0;
      for I := 0 to 1 do
      begin
        if I <> 0 then R := R xor K0 else R := R xor K1;
        for J := 0 to 3 do
        begin
          B := R shr 24;
          if B and $80 <> 0 then G2 := (B shl 1 xor $014D) and $FF
            else G2 := B shl 1 and $FF;
          if B and 1 <> 0 then G3 := (B shr 1 and $7F) xor $014D shr 1 xor G2
            else G3 := (B shr 1 and $7F) xor G2;
          R := R shl 8 xor G3 shl 24 xor G2 shl 16 xor G3 shl 8 xor B;
        end;
      end;
      Result := R;
    end;

    function F32(X: Integer; K: array of Integer): Integer;
```

```
  var
    A, B, C, D: Integer;
  begin
    A := X and $FF;
    B := X shr  8 and $FF;
    C := X shr 16 and $FF;
    D := X shr 24;
    if Size = 32 then
    begin
      A := Twofish_8x8[1, A] xor K[3] and $FF;
      B := Twofish_8x8[0, B] xor K[3] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[3] shr 16 and $FF;
      D := Twofish_8x8[1, D] xor K[3] shr 24;
    end;
    if Size >= 24 then
    begin
      A := Twofish_8x8[1, A] xor K[2] and $FF;
      B := Twofish_8x8[1, B] xor K[2] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[2] shr 16 and $FF;
      D := Twofish_8x8[0, D] xor K[2] shr 24;
    end;
    A := Twofish_8x8[0, A] xor K[1] and $FF;
    B := Twofish_8x8[1, B] xor K[1] shr  8 and $FF;
    C := Twofish_8x8[0, C] xor K[1] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[1] shr 24;

    A := Twofish_8x8[0, A] xor K[0] and $FF;
    B := Twofish_8x8[0, B] xor K[0] shr  8 and $FF;
    C := Twofish_8x8[1, C] xor K[0] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[0] shr 24;

    Result := Twofish_Data[0, A] xor Twofish_Data[1, B] xor
              Twofish_Data[2, C] xor Twofish_Data[3, D];
  end;

var
  I,J,A,B: Integer;
  E,O: array[0..3] of Integer;
  K: array[0..7] of Integer;
begin
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  if Size <= 16 then Size := 16 else
    if Size <= 24 then Size := 24
      else Size := 32;
  J := Size shr 3 - 1;
  for I := 0 to J do
  begin
    E[I] := K[I shl 1];
    O[I] := K[I shl 1 + 1];
    BoxKey[J].L := Encode(E[I], O[I]);
    Dec(J);
  end;
  J := 0;
  for I := 0 to 19 do
  begin
    A := F32(J, E);
    B := ROL(F32(J + $01010101, O), 8);
    SubKey[I shl 1] := A + B;
    B := A + B shr 1;
    SubKey[I shl 1 + 1] := ROL(B, 9);
    Inc(J, $02020202);
  end;
end;

procedure DoXOR(D, S: PIntArray; Value: LongWord);
var
  I: LongWord;
begin
```

```
    Value := (Value and $FF) * $01010101;
    for I := 0 to 63 do D[I] := S[I] xor Value;
  end;

  procedure SetupBox128;
  var
    L: array[0..255] of Byte;
    A,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L);
    A := BoxKey[0].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 8);
    A := BoxKey[0].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L shr 16);
    A := BoxKey[0].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 24);
    A := BoxKey[0].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, L[I]] xor A];
  end;

  procedure SetupBox192;
  var
    L: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L);
    A := BoxKey[0].A;
    B := BoxKey[1].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L ,@Twofish_8x8[0], BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

  procedure SetupBox256;
  var
    L: array[0..255] of Byte;
    K: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L);
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L);
    A := BoxKey[0].A;
    B := BoxKey[1].A;
```

```
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0], BoxKey[3].L shr 8);
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0],BoxKey[3].L shr 16);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L shr 24);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

begin
  InitBegin(Size);
  SubKey := User;
  Box    := @SubKey[40];
  SetupKey;
  if Size = 16 then SetupBox128 else
    if Size = 24 then SetupBox192
      else SetupBox256;
  InitEnd(IVector);
end;

class procedure TCipher_Shark.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 112;
end;

class function TCipher_Shark.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    0D9h,065h,021h,0AAh,0C0h,0C3h,084h,060h
        DB    09Dh,0CEh,01Fh,08Bh,0FBh,0ABh,018h,03Fh
        DB    0A1h,021h,0ACh,0F8h,053h,049h,0C0h,06Fh
        DB    027h,03Ah,089h,015h,0D3h,07Ah,0E9h,00Bh
end;

{$IFDEF VER_D4H} // >= D4
  {.$DEFINE Shark64}
{$ENDIF}

type
  PInt64        = ^TInt64;
{$IFDEF Shark64}
  TInt64        = Int64;
{$ELSE}
  TInt64        = packed record
                    L,R: Integer;
```

```
                end;
{$ENDIF}

  PInt64Array = ^TInt64Array;
  TInt64Array = array[0..1023] of TInt64;

{$IFDEF Shark64}
  TShark_Data = array[0..7, 0..255] of Int64;
{$ENDIF}

procedure TCipher_Shark.Encode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  L,R: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
         TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CE)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SE[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    T := Shark_CE[0, R shr 23 and $1FE] xor
         Shark_CE[1, R shr 15 and $1FE] xor
         Shark_CE[2, R shr  7 and $1FE] xor
         Shark_CE[3, R shl  1 and $1FE] xor
         Shark_CE[4, L shr 23 and $1FE] xor
         Shark_CE[5, L shr 15 and $1FE] xor
         Shark_CE[6, L shr  7 and $1FE] xor
         Shark_CE[7, L shl  1 and $1FE];
    R := Shark_CE[0, R shr 23 and $1FE or 1] xor
         Shark_CE[1, R shr 15 and $1FE or 1] xor
         Shark_CE[2, R shr  7 and $1FE or 1] xor
         Shark_CE[3, R shl  1 and $1FE or 1] xor
         Shark_CE[4, L shr 23 and $1FE or 1] xor
         Shark_CE[5, L shr 15 and $1FE or 1] xor
         Shark_CE[6, L shr  7 and $1FE or 1] xor
```

```
          Shark_CE[7, L shl  1 and $1FE or 1];
      L := T;
    end;
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    L := LongWord(Shark_SE[L shr 24        ]) shl 24 xor
         LongWord(Shark_SE[L shr 16 and $FF]) shl 16 xor
         LongWord(Shark_SE[L shr  8 and $FF]) shl  8 xor
         LongWord(Shark_SE[L        and $FF]);
    R := LongWord(Shark_SE[R shr 24        ]) shl 24 xor
         LongWord(Shark_SE[R shr 16 and $FF]) shl 16 xor
         LongWord(Shark_SE[R shr  8 and $FF]) shl  8 xor
         LongWord(Shark_SE[R        and $FF]);
    PInt64(Data).L := L xor K[0];
    PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Decode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  R,L: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  Inc(K, 7);
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CD)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CD)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CD)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CD)[3, D shr 32 and $FF] xor

         TShark_Data(Shark_CD)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CD)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CD)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CD)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SD[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SD[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SD[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SD[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SD[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SD[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SD[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SD[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  Inc(PInteger(K), 14);
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    T := Shark_CD[0, R shr 23 and $1FE] xor
         Shark_CD[1, R shr 15 and $1FE] xor
         Shark_CD[2, R shr  7 and $1FE] xor
```

```
          Shark_CD[3, R shl  1 and $1FE] xor
          Shark_CD[4, L shr 23 and $1FE] xor
          Shark_CD[5, L shr 15 and $1FE] xor
          Shark_CD[6, L shr  7 and $1FE] xor
          Shark_CD[7, L shl  1 and $1FE];
    R := Shark_CD[0, R shr 23 and $1FE or 1] xor
          Shark_CD[1, R shr 15 and $1FE or 1] xor
          Shark_CD[2, R shr  7 and $1FE or 1] xor
          Shark_CD[3, R shl  1 and $1FE or 1] xor
          Shark_CD[4, L shr 23 and $1FE or 1] xor
          Shark_CD[5, L shr 15 and $1FE or 1] xor
          Shark_CD[6, L shr  7 and $1FE or 1] xor
          Shark_CD[7, L shl  1 and $1FE or 1];
    L := T;
  end;
  L := L xor K[0];
  R := R xor K[1];
  Inc(PInteger(K), 2);
  L := Integer(Shark_SD[L shr 24        ]) shl 24 xor
       Integer(Shark_SD[L shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[L shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[L        and $FF]);
  R := Integer(Shark_SD[R shr 24        ]) shl 24 xor
       Integer(Shark_SD[R shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[R shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[R        and $FF]);
  PInt64(Data).L := L xor K[0];
  PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Init(const Key; Size: Integer; IVector: Pointer);
var
  Log, ALog: array[0..255] of Byte;

  procedure InitLog;
  var
    I, J: Word;
  begin
    ALog[0] := 1;
    for I := 1 to 255 do
    begin
      J := ALog[I-1] shl 1;
      if J and $100 <> 0 then J := J xor $01F5;
      ALog[I] := J;
    end;
    for I := 1 to 254 do Log[ALog[I]] := I;
  end;

  function Transform(A: TInt64): TInt64;
  type
    TInt64Rec = packed record
                  Lo, Hi: Integer;
                end;

    function Mul(A, B: Integer): Byte;
    begin
      Result := ALog[(Log[A] + Log[B]) mod 255];
    end;

  var
    I,J: Byte;
    K,T: array[0..7] of Byte;
  begin
{$IFDEF Shark64}
    Move(TInt64Rec(A).Hi, K[0], 4);
    Move(TInt64Rec(A).Lo, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
{$ELSE}
```

```
    Move(A.R, K[0], 4);
    Move(A.L, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
{$ENDIF}
    for I := 0 to 7 do
    begin
      T[I] := Mul(Shark_I[I, 0], K[0]);
      for J := 1 to 7 do T[I] := T[I] xor Mul(Shark_I[I, J], K[J]);
    end;
{$IFDEF Shark64}
    Result := T[0];
    for I := 1 to 7 do Result := Result shl 8 xor T[I];
{$ELSE}
    Result.L := T[0];
    Result.R := 0;
    for I := 1 to 7 do
    begin
      Result.R := Result.R shl 8 or Result.L shr 24;
      Result.L := Result.L shl 8 xor T[I];
    end;
{$ENDIF}
  end;

  function Shark(D: TInt64; K: PInt64): TInt64;
  var
    R,T: Integer;
  begin
{$IFDEF Shark64}
    for R := 0 to 4 do
    begin
      D := D xor K^; Inc(K);
      D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
           TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
           TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
           TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
           TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
           TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
           TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
           TShark_Data(Shark_CE)[7, D        and $FF];
    end;
    D := D xor K^; Inc(K);
    D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
         (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
         (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
         (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
         (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
         (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
         (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
         (Int64(Shark_SE[D        and $FF]));
    Result := D xor K^;
{$ELSE}
    for R := 0 to 4 do
    begin
      D.L := D.L xor K.L;
      D.R := D.R xor K.R;
      Inc(K);
      T   := Shark_CE[0, D.R shr 23 and $1FE] xor
             Shark_CE[1, D.R shr 15 and $1FE] xor
             Shark_CE[2, D.R shr  7 and $1FE] xor
             Shark_CE[3, D.R shl  1 and $1FE] xor
             Shark_CE[4, D.L shr 23 and $1FE] xor
             Shark_CE[5, D.L shr 15 and $1FE] xor
             Shark_CE[6, D.L shr  7 and $1FE] xor
             Shark_CE[7, D.L shl  1 and $1FE];

      D.R := Shark_CE[0, D.R shr 23 and $1FE or 1] xor
             Shark_CE[1, D.R shr 15 and $1FE or 1] xor
             Shark_CE[2, D.R shr  7 and $1FE or 1] xor
             Shark_CE[3, D.R shl  1 and $1FE or 1] xor
```

```
              Shark_CE[4, D.L shr 23 and $1FE or 1] xor
              Shark_CE[5, D.L shr 15 and $1FE or 1] xor
              Shark_CE[6, D.L shr  7 and $1FE or 1] xor
              Shark_CE[7, D.L shl  1 and $1FE or 1];
      D.L := T;
    end;
    D.L := D.L xor K.L;
    D.R := D.R xor K.R;
    Inc(K);
    D.L := Integer(Shark_SE[D.L shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.L shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.L shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.L        and $FF]);
    D.R := Integer(Shark_SE[D.R shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.R shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.R shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.R        and $FF]);
    Result.L := D.L xor K.L;
    Result.R := D.R xor K.R;
{$ENDIF}
  end;

var
  T: array[0..6] of TInt64;
  A: array[0..6] of TInt64;
  K: array[0..15] of Byte;
  I,J,R: Byte;
  E,D: PInt64Array;
  L: TInt64;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  InitLog;
  E := User;
  D := @E[7];
  Move(Shark_CE[0], T, SizeOf(T));
  T[6] := Transform(T[6]);
  I := 0;
{$IFDEF Shark64}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R] := K[I and $F];
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R] := A[R] shl 8 or K[I and $F];
    end;
  end;
  E[0] := A[0] xor Shark(0, @T);
  for R := 1 to 6 do E[R] := A[R] xor Shark(E[R - 1], @T);
{$ELSE}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R].L := K[I and $F];
    A[R].R := 0;
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R].R := A[R].R shl 8 or A[R].L shr 24;
      A[R].L := A[R].L shl 8 or K[I and $F];
    end;
  end;
  L.L := 0;
  L.R := 0;
  L := Shark(L, @T);
  E[0].L := A[0].L xor L.L;
```

```
  E[0].R := A[0].R xor L.R;
  for R := 1 to 6 do
  begin
    L := Shark(E[R - 1], @T);
    E[R].L := A[R].L xor L.L;
    E[R].R := A[R].R xor L.R;
  end;
{$ENDIF}

  E[6] := Transform(E[6]);
  D[0] := E[6];
  D[6] := E[0];
  for R := 1 to 5 do D[R] := Transform(E[6-R]);

  FillChar(Log, SizeOf(Log), 0);
  FillChar(ALog, SizeOf(ALog), 0);
  FillChar(T, SizeOf(T), 0);
  FillChar(A, SizeOf(A), 0);
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

class procedure TCipher_Square.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 9 * 4 * 2 * SizeOf(LongWord);
end;

class function TCipher_Square.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    043h,09Ch,0A6h,0C4h,067h,0E8h,02Eh,047h
         DB    022h,095h,066h,085h,006h,039h,06Ah,0C9h
         DB    018h,021h,020h,0F7h,044h,036h,0F1h,061h
         DB    07Dh,014h,090h,0B1h,0A9h,068h,056h,0C7h
end;

procedure TCipher_Square.Encode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := User;
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);
  for I := 0 to 6 do
  begin
    AA := Square_TE[0, A        and $FF] xor
          Square_TE[1, B        and $FF] xor
          Square_TE[2, C        and $FF] xor
          Square_TE[3, D        and $FF] xor Key[0];
    BB := Square_TE[0, A shr  8 and $FF] xor
          Square_TE[1, B shr  8 and $FF] xor
          Square_TE[2, C shr  8 and $FF] xor
          Square_TE[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TE[0, A shr 16 and $FF] xor
          Square_TE[1, B shr 16 and $FF] xor
          Square_TE[2, C shr 16 and $FF] xor
          Square_TE[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TE[0, A shr 24        ] xor
          Square_TE[1, B shr 24        ] xor
```

```
          Square_TE[2, C shr 24        ] xor
          Square_TE[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);

    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SE[A        and $FF])        xor
                        LongWord(Square_SE[B        and $FF]) shl  8 xor
                        LongWord(Square_SE[C        and $FF]) shl 16 xor
                        LongWord(Square_SE[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SE[A shr  8 and $FF])        xor
                        LongWord(Square_SE[B shr  8 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SE[A shr 16 and $FF])        xor
                        LongWord(Square_SE[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SE[A shr 24       ])        xor
                        LongWord(Square_SE[B shr 24       ]) shl  8 xor
                        LongWord(Square_SE[C shr 24       ]) shl 16 xor
                        LongWord(Square_SE[D shr 24       ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Decode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := @PIntArray(User)[9 * 4];
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);

  for I := 0 to 6 do
  begin
    AA := Square_TD[0, A        and $FF] xor
          Square_TD[1, B        and $FF] xor
          Square_TD[2, C        and $FF] xor
          Square_TD[3, D        and $FF] xor Key[0];
    BB := Square_TD[0, A shr  8 and $FF] xor
          Square_TD[1, B shr  8 and $FF] xor
          Square_TD[2, C shr  8 and $FF] xor
          Square_TD[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TD[0, A shr 16 and $FF] xor
          Square_TD[1, B shr 16 and $FF] xor
          Square_TD[2, C shr 16 and $FF] xor
          Square_TD[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TD[0, A shr 24        ] xor
          Square_TD[1, B shr 24        ] xor
          Square_TD[2, C shr 24        ] xor
          Square_TD[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);
    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SD[A        and $FF])        xor
                        LongWord(Square_SD[B        and $FF]) shl  8 xor
                        LongWord(Square_SD[C        and $FF]) shl 16 xor
                        LongWord(Square_SD[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SD[A shr  8 and $FF])        xor
                        LongWord(Square_SD[B shr  8 and $FF]) shl  8 xor
```

```
                        LongWord(Square_SD[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SD[A shr 16 and $FF])        xor
                        LongWord(Square_SD[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SD[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SD[A shr 24        ])        xor
                        LongWord(Square_SD[B shr 24        ]) shl  8 xor
                        LongWord(Square_SD[C shr 24        ]) shl 16 xor
                        LongWord(Square_SD[D shr 24        ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Init(const Key; Size: Integer; IVector: Pointer);
type
  PSquare_Key = ^TSquare_Key;
  TSquare_Key = array[0..8, 0..3] of LongWord;
var
  E,D: PSquare_Key;
  T,I: Integer;
begin
  InitBegin(Size);
  E := User;
  D := User; Inc(D);
  Move(Key, E^, Size);
  for T := 1 to 8 do
  begin
    E[T, 0] := E[T -1, 0] xor ROR(E[T -1, 3], 8) xor 1 shl (T - 1); D[8 -T, 0]
:= E[T, 0];
    E[T, 1] := E[T -1, 1] xor E[T, 0];                               D[8 -T, 1]
:= E[T, 1];
    E[T, 2] := E[T -1, 2] xor E[T, 1];                               D[8 -T, 2]
:= E[T, 2];
    E[T, 3] := E[T -1, 3] xor E[T, 2];                               D[8 -T, 3]
:= E[T, 3];
    for I := 0 to 3 do
      E[T -1, I] :=     Square_PHI[E[T -1, I]        and $FF]      xor
                    ROL(Square_PHI[E[T -1, I] shr  8 and $FF],  8) xor
                    ROL(Square_PHI[E[T -1, I] shr 16 and $FF], 16) xor
                    ROL(Square_PHI[E[T -1, I] shr 24        ], 24);
  end;
  D[8] := E[0];
  InitEnd(IVector);
end;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386


procedure FindVirtualMethodAndChange(AClass: TClass; MethodAddr, NewAddress:
Pointer);
type
  PPointer = ^Pointer;
const
  PageSize = SizeOf(Pointer);
var
  Table: PPointer;
  SaveFlag: DWORD;
begin
  Table := PPointer(AClass);
  while Table^ <> MethodAddr do Inc(Table);
  if VirtualProtect(Table, PageSize, PAGE_EXECUTE_READWRITE, @SaveFlag) then
  try
    Table^ := NewAddress;
  finally
    VirtualProtect(Table, PageSize, SaveFlag, @SaveFlag);
  end;
end;
  {$ENDIF}
{$ENDIF}
```

```
{$IFDEF VER_D3H}
procedure ModuleUnload(Module: Integer);
var
  I: Integer;
begin
  if IsObject(FCipherList, TStringList) then
    for I := FCipherList.Count-1 downto 0 do
      if FindClassHInstance(TClass(FCipherList.Objects[I])) = Module then
        FCipherList.Delete(I);
end;
{$ENDIF}

initialization
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
  if CPUType <= 3 then  // CPU <= 386
  begin
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Encode,
                                                 @TCipher_Blowfish.Encode386);
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Decode,
                                                 @TCipher_Blowfish.Decode386);
  end;
  {$ENDIF}
{$ENDIF}
{$IFDEF VER_D3H}
  AddModuleUnloadProc(ModuleUnload);
{$ENDIF}
{$IFNDEF ManualRegisterClasses}
  RegisterCipher(TCipher_3Way, '', '');
  RegisterCipher(TCipher_Blowfish, '', '');
  RegisterCipher(TCipher_Gost, '', '');
  RegisterCipher(TCipher_IDEA, '', 'не комерційний');
  RegisterCipher(TCipher_Q128, '', '');
  RegisterCipher(TCipher_SAFER_K40, 'SAFER-K40', '');
  RegisterCipher(TCipher_SAFER_SK40, 'SAFER-SK40', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K64, 'SAFER-K64', '');
  RegisterCipher(TCipher_SAFER_SK64, 'SAFER-SK64', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K128, 'SAFER-K128', '');
  RegisterCipher(TCipher_SAFER_SK128, 'SAFER-SK128', 'Keyscheduling');
  RegisterCipher(TCipher_SCOP, '', '');
  RegisterCipher(TCipher_Shark, '', '');
  RegisterCipher(TCipher_Square, '', '');
  RegisterCipher(TCipher_TEA, 'TEA', '');
  RegisterCipher(TCipher_TEAN, 'TEA розширений', '');
  RegisterCipher(TCipher_Twofish, '', '');
{$ENDIF}
finalization
{$IFDEF VER_D3H}
  RemoveModuleUnloadProc(ModuleUnload);
{$ENDIF}
  FCipherList.Free;
  FCipherList := nil;
end.
```