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 формування профілів захисту хмарних сервісів”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Петренюк В.І.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

Яценку Денису Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Ященко Д.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Ященко Д.Р. Дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

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

Метою розробки є дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів.

Об'єктом дослідження є процес формування профілів захисту хмарних сервісів.

Предметом дослідження є методи формування профілів захисту хмарних сервісів.

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

Результат роботи – програмна реалізація системи формування профілів захисту хмарних сервісів.

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

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

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

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, профілі захисту, хмарні сервіси

ABSTRACT

Yashchenko D.R. Research and software implementation of the system for forming cloud service protection profiles. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this final qualification work for the second (master's) level of higher education, software was developed, which is intended for the system of forming profiles for the protection of cloud services.

The purpose of the development is the research and software implementation of the system for forming cloud service protection profiles.

The object of the study is the process of forming cloud service protection profiles.

The subject of the study is the methods of forming cloud service protection profiles.

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

The result of the work is the software implementation of the system for forming cloud service protection profiles.

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 RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, protection profiles, cloud services

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АБС	–	автоматизована банківська система
АІС	–	автоматизованих інформаційних системах
БЗ	–	база знань
ЕС	–	експертна система
ЗК	–	загальні критерії
ІБ	–	інформаційна безпека
ІС	–	інформаційна система
КБ	–	комерційний банк
КС	–	конкурентоспроможність
НЛ	–	системи нечіткої логіки
НМ	–	нейронні мережі
ПЗ	–	профіль захисту
ПК	–	програмний комплекс
СЗІ	–	система захисту інформації

ВСТУП

Актуальність теми. В останні роки в банківській діяльності загострилася проблема забезпечення безпеки даних. Вона містить у собі кілька аспектів. По-перше, це гнучка, багаторівнева й надійна регламентація повноважень користувачів – цінність банківської інформації висуває особливі вимоги до захисту даних від несанкціонованого доступу, у тому числі, до контролю керування процесами, що змінюють стан даних. По-друге, важливим аспектом є наявність засобів для підтримки цілісності й несуперечності даних; подібні засоби мають на увазі можливість здійснення контролю введення даних, підтримки й контролю зв'язків між даними, а також уведення й модифікації даних у режимі транзакцій – набір операцій, що забезпечують підтримку погодженості даних. По-третє, необхідна присутність у системі багатофункціональних процедур архівації, відновлення й моніторингу даних при програмних і апаратних збоях.

Забезпечення безпеки інформаційних банківських систем являє собою комплексну проблему, що вирішується в напрямках удосконалювання правового регулювання застосування інформаційних технологій, удосконалювання методів і засобів їхньої розробки, розвитку системи сертифікації, забезпечення відповідних організаційно-технічних умов експлуатації. Ключовим аспектом рішення проблеми безпеки є вироблення системи вимог, критеріїв і показників для оцінки рівня безпеки інформаційних технологій.

ДСТ ІСО/МЕК 15408 визначає критерії, за яких історично закріпилася назва "Загальні критерії" (ЗК). ЗК призначені для використання інформаційних технологій як основу при оцінці характеристик безпеки продуктів і систем. Установлюючи загальну базу критеріїв, ЗК роблять результати оцінки безпеки значимими для більше широкої аудиторії.

Сукупність вимог безпеки, узятих з ЗК або сформульованих у явному виді,

представляється у вигляді профілю захисту, оцінка й обґрунтування якого виконується відповідно до критеріїв оцінки, що втримується в частині 3 ЗК. Метою такої оцінки є демонстрація того, що профіль повний, несуперечливий, технічно правильний і придатний для використання при викладі вимог до об'єкта оцінки, передбачуваному для оцінки.

Обґрунтування ж містить у собі наступне:

а) логічне обґрунтування цілей безпеки, що демонструє, що викладені цілі безпеки зіставлені з усіма аспектами середовища безпеки;

б) логічне обґрунтування вимог безпеки, що демонструє, що сукупність вимог безпеки придатна для досягнення цілей безпеки й порівнянна з ними.

Проте, відсутність методології економічного обґрунтування й оцінки профілю захисту в цей час приводить до відсутності прагнення до впровадження ЗК. Розробка моделі обґрунтування, що опирається на економічні показники діяльності, буде стимулювати впровадження ЗК у різні галузеві сфери, зокрема, у сферу банківських інформаційних технологій. Це дозволить значно підвищити рівень безпеки банківських інформаційних систем, збільшить довіру до них як з боку користувачів (банківських організацій), так і сторони кінцевих споживачів банківського продукту.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів.

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

– Огляд існуючих систем формування профілів захисту хмарних сервісів.

– Дослідження системи формування профілів захисту хмарних сервісів.

– Програмна реалізація системи формування профілів захисту хмарних сервісів.

Об’єктом дослідження є процес формування профілів захисту хмарних сервісів.

Предметом дослідження є методи формування профілів захисту хмарних сервісів.

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

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

– Удосконалено метод формування профілів захисту хмарних сервісів.

– Розроблено вітчизняний продукт формування профілів захисту хмарних сервісів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі формування профілів захисту хмарних сервісів.

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

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

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

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Комплексна експертна система керування інформаційною безпекою є програмним продуктом, призначеним для рішення завдань керування безпекою в великих територіально-розподілених автоматизованих інформаційних системах (АІС), і покликана полегшувати завдання контролю центральними структурами рівня забезпечення інформаційної безпеки на місцях.

Типовий пакет програмних засобів ЕС містить у собі два програмних комплекси: «Аналіз» і «Контроль». Основною функцією першого комплексу є забезпечення формування по можливості найбільш повного набору вимог до безпеки всіх складових АІС організації, бізнес і технологічні процеси. Це завдання вирішується шляхом побудови моделей погроз, моделей подій ризиків, моделей і профілів захисту (ПЗ), моделей комплексів заходів, розрахунку початкових і залишкових (після виконання того або іншого комплексу заходів) ризиків.

Програмний комплекс «Контроль», у свою чергу, складається із двох частин – програмного комплексу (ПрКс) "Центр" і ПрКс "Регіон". Перший призначений для:

– розробки профілів захисту (ПрЗах);

– підготовки й розсилання ПрЗах за допомогою електронної пошти по підконтрольних регіональних частинах АІС;

– для автоматизованого збору звітності про виконання вимог безпеки в регіонах; для оцінки ризиків невиконання вимог безпеки в АІС;

– для ідентифікації вузьких місць у захисті.

Другий – для одержання профілів захисту (ПрЗах) у регіонах; для автоматизації ведення в регіонах звітності про виконання ПрЗах і для відсилання цієї звітності по електронній пошті в Центр, де вона повинна в автоматичному режимі оброблятися ПрКс "Центр".

Таким чином, програмний комплекс «Аналіз» покликаний відігравати допоміжну роль у рішенні завдань керування ІБ, а саме, забезпечувати повноцінний всебічний аналіз, вирішує сформулювати обґрунтований набір цілей безпеки, обґрунтувати політику безпеки, гарантувати повноту вимог безпеки, контроль виконання яких потрібно здійснювати. На практиці, часто вимоги для ПрКс «Контроль» формуються без використання комплексу «Аналіз». При цьому працюючі із ПрКс "Центр" експерти керуються власним досвідом і вимогами, сформульованими в офіційних документах ДСТЗІ СБУ України, інших нормативних документах, наказах, посадових інструкціях, руководствах і інших документах, у яких закріплена політика безпеки АІС.

Однією з допоміжних функцій, можливість виконання якої закладено як у комплексі «Аналіз», так і в комплексі «Центр», є побудова профілів захисту на підставі вимог сформульованих у ДСТ ІСО/МЕК 15408-2002. Основна відмінність у методологіях побудови ПрЗах із використанням ПрКс «Аналіз» і ПрКс «Центр» полягає в тому, що в першому комплексі побудова ПрЗах здійснюється через побудову моделей погроз, де кожне з вимог зафіксованих у ДСТ ІСО/МЕК 15408-2002 розглядається через призму погроз, які при його дотриманні можуть бути в тому або іншому ступені відбиті, і через аналіз ризиків, які можуть бути в результаті дотримання цієї вимоги знижені. При використанні ПрКс «Контроль» вимоги, що втримуються в ДСТ ІСО/МЕК 15408-2002 формуються у вигляді ПрЗах експертами, що опираються на наявні в них знання, досвід і на дані аналізу попередньо проведеного з використанням ПрКс "Аналіз". Обидва програмних комплекси дозволяють фіксувати вимоги із вказівкою додаткових умов, взаємозв'язків і взаємозалежностей вимог. Обидва дозволяють, будувати типові профілі захисту й використовувати їх як шаблони для побудови нових профілів. Обидва дозволяють доповнювати вимоги, сформовані на основі ДСТ ІСО/МЕК 15408-2002 вимогами дотичних адміністративних мір (організаційних мір, підбора, підготовки, забезпечення персоналу), мер процедурного й фізичного контролю.

1.2 Область застосування

Необхідним атрибутом успішного функціонування більшості державних і приватних структур є їх власна інформаційна безпека. Залежно від роду діяльності підприємства, організації й т.д. необхідний рівень захищеності може варіювати в широких межах, однак для його досягнення однозначно необхідне використання відповідного комплексу спеціальних засобів і заходів.

У сучасний час оцінка рівня інформаційної захищеності об'єкта (деякої території, будинку, приміщення, технічного засобу або їхнього комплексу й т.п.), його зіставлення з об'єктивно необхідним рівнем, а у випадку їхньої невідповідності – підбор оптимального комплексу засобів і заходів щодо підвищення інформаційної безпеки являє собою досить складне наукомістке завдання, що вимагає від виконавця наявності не тільки глибоких предметних знань і практичного досвіду, але й навіть інтуїції. У зв'язку із цим його рішення було й залишається в значній мірі прерогативою обмеженого кола фахівців. Це завдання, з одного боку, характеризується високою вартістю процедури аналізу й оптимізації інформаційної захищеності, з іншого боку – значною залежністю кінцевого результату від суб'єктивного фактора, що, мабуть, являє собою негативний момент. Разом з тим, аналіз і оптимізація інформаційної захищеності об'єкта по класичному визначенню ставляться до числа важкоструктуруємих і важкоформалізуємих завдань. У зв'язку із цим ефективним інструментом для підтримки її рішення може бути експертна система (ЕС) або інший інтелектуальний комп'ютерний засіб.

Предметні знання експертів акумулюються в базі знань ЕС і можуть бути використані в потрібний час або тиражовані. ЕС по оцінці рівня захисту інформації повинна дозволити протиріччя між зростаючою потребою залучення кваліфікованих експертів по інформаційній безпеці для організації захисту на об'єктах і їхній обмеженій кількості.

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

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

У рамках завдання поставлених при проведенні магістерської роботи, проведемо дослідження існуючих експертних систем, що визначають ступінь захищеності інформації.

Експертна система розроблена в НВП "Фотон" НТУУ "КПІ"

У НВП "Фотон" НТУУ "КПІ" розроблена ЕС, призначена для використання службами (підрозділами) державних, комерційних, суспільно-політичних організацій, відповідальними за захист інформації з обмеженим доступом, зокрема, від несанкціонованого доступу з використанням технічних засобів нападу, застосовуваних для добування інформації із природного або штучно створеним каналам витоку.

ЕС дозволяє в діалоговому режимі з користувачем виявити характеристики об'єкта, інформація в межах якого повинна бути захищена від витоку, і, ґрунтуючись на базі знань, отриманих від експертів, оцінити погрозу (імовірність витоку) інформації й запропонувати організаційні міри й технічні засоби захисту, що дозволяють користувачеві вибрати ті з них, які підвищують ефективність захисту (знижують імовірність витоку інформації) до необхідного рівня, а також оцінити вартість витрат на захист інформації. У процесі розробки ЕС виконана систематизація даних про засоби технічної розвідки, створена база знань експертів по захисту інформації, розроблені алгоритми оцінки рівня погроз і реалізація відповідного програмного забезпечення, апаратний і програмний захист від несанкціонованого копіювання ЕС, її бази знань і даних про об'єкти,

забезпечення апаратними й програмними засобами тільки санкціонованого доступу до роботи з ЕС і даними про окремі об'єкти.

ЕС реалізується на базі IBM PC і орієнтована на роботу в середовищі Windows 3.xx і вище. Дружній інтерфейс мінімізує вимоги по попередньому навчанню користувача. На рисунку наведено одне з діалогових вікон, пропонованих ЕС користувачеві.

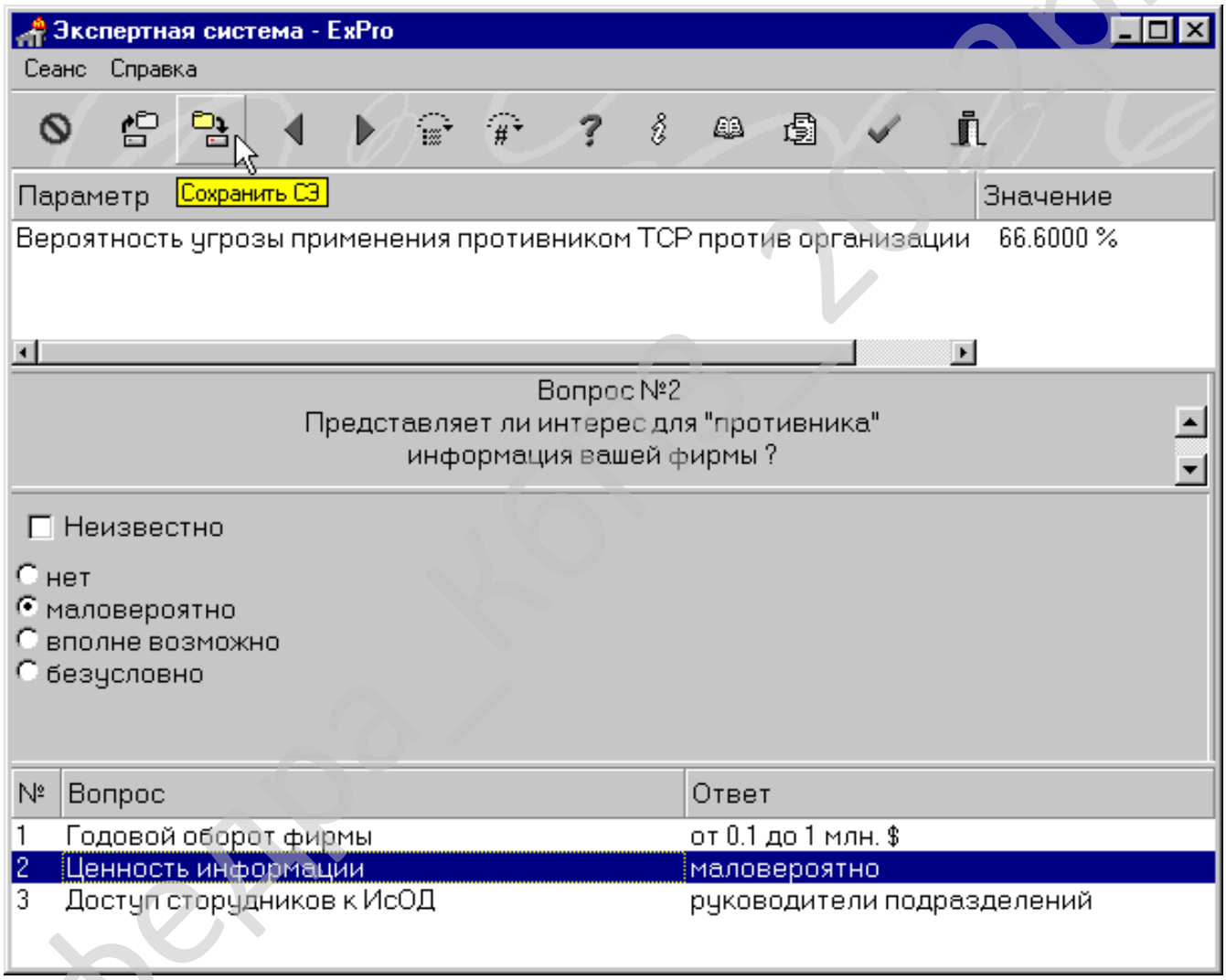


Рисунок 2.1 – Інтерфейс користувача експертної системи розробленої в НВП “Фотон” НТУУ “КПІ

ЕС володіє також можливостями інформаційно-довідкової системи по засобах нападу й захисту, дозволяє користувачеві вводити зміни в дані по засобах

напади й захисти, доповнювати й коректувати базу знань, будучи по суті автоматизованим робочим місцем експерта по безпеці інформації на об'єкті.

Spectrum

Ціль програмного продукту – аналіз уразливостей інформаційної безпеки компонентів автоматизованих систем (таких як сервера баз даних, міжмережні екрани, маршрутизатори, сервера додатків, термінальні, поштові, WEB, робочі станції) від зловмисника з метою усунення цих уразливостей, оптимізації й підвищення надійності.

Аналіз локальної мережі – це комплексна діагностика платформи функціонування інформаційної системи і її складової частини – комп'ютерної мережі з метою локалізації дефектів і "вузьких місць", що характеризують загальний стан, а також дає можливість оцінити кількісні характеристики і якість обслуговування.

Внутрішній аналіз ресурсів локальної мережі:

– вивчення особливостей апаратних і програмних ресурсів робочих станцій і серверів локальної мережі;

– стан носіїв інформації, їх відказостійкість;

– наявність систем резервного копіювання інформації;

– аналіз розподілу прав доступу й можливостей одержання інформації з публічних або вилучених мереж.

Все різноманіття засобів, застосовуваних для моніторингу й аналізу обчислювальних мереж, можна розділити на кілька великих класів:

– Системи керування мережею (NetworkManagementSystems) – централізовані програмні системи, які збирають дані про стан вузлів і комунікаційних пристроїв мережі, а також дані про трафік, що циркулює в мережі. Ці системи не тільки здійснюють моніторинг і аналіз мережі, але й виконують в автоматичному або напівавтоматичному режимі дії по керуванню мережею – включення й відключення портів пристроїв, зміна параметрів мостів адресних таблиць мостів, комутаторів і маршрутизаторів і т.п. Прикладами систем

керування можуть служити популярні системи HPOpenView, SunNetManager, IBMNetView.

– Засоби керування системою (SystemManagement). Засоби керування системою часто виконують функції, аналогічні функціям систем керування, але стосовно інших об'єктів. У першому випадку об'єктом керування є програмне й апаратне забезпечення комп'ютерів мережі, а в другому – комунікаційне встаткування. Разом з тим, деякі функції цих двох видів систем керування можуть дублюватися, наприклад, засобу керування системою можуть виконувати найпростіший аналіз мережного трафіку.

– Убудовані системи діагностики й керування (Embeddedsystems). Ці системи виконуються у вигляді програмно-апаратних модулів, установлюваних у комунікаційне встаткування, а також у вигляді програмних модулів, убудованих в операційні системи. Вони виконують функції діагностики й керування тільки одним пристроєм, і в цьому їхню основну відмінність від централізованих систем керування. Прикладом засобів цього класу може служити модуль керування концентратором Distrebuted 5000, що реалізує функції автосегментації портів при виявленні несправностей, приписування портів внутрішнім сегментам концентратора й деякі інші. Як правило, убудовані модулі керування "по сумісництву" виконують роль SNMP-Агентів, що поставляють дані про стан пристрої для систем керування.

– Аналізатори протоколів (Protocolanalyzers). Являють собою програмні або апаратно-програмні системи, які обмежуються на відміну від систем керування лише функціями моніторингу й аналізу трафіку в мережах. Гарний аналізатор протоколів може захоплювати й декодувати пакети великої кількості протоколів, застосовуваних у мережах – звичайно трохи десятків. аналізатори протоколів дозволяють установити деякі логічні умови для захвата окремих пакетів і виконують повне декодування захоплених пакетів, тобто показують у зручній для фахівця формі вкладеність пакетів протоколів різних рівнів друг у друга з розшифровкою змісту окремих полів кожного пакета.

– Устаткування для діагностики й сертифікації кабельних систем. Умовно це встаткування можна поділити на чотири основні групи: мережні монітори,

прилади для сертифікації кабельних систем, кабельні сканери й тестери (мультиметри).

– Мережні монітори (називані також мережними аналізаторами) призначені для тестування кабелів різних категорій. Варто розрізняти мережні монітори й аналізатори протоколів. Мережні монітори збирають дані тільки про статистичні показники трафіку – середньої інтенсивності загального трафіку мережі, середній інтенсивності потоку пакетів з певним типом помилки й т.п.

– Призначення пристроїв для сертифікації кабельних систем, безпосередньо виходить з їхньої назви. Сертифікація виконується відповідно до вимог одного з міжнародних стандартів на кабельні системи.

– Кабельні сканери використовуються для діагностики мідних кабельних систем.

– Тестери призначені для перевірки кабелів на відсутність фізичного розриву.

– Експертні системи. Цей вид систем акумулює людські знання про виявлення причин аномальної роботи мереж і можливих способів приведення мережі в працездатний стан. Експертні системи часто реалізуються у вигляді окремих підсистем різних засобів моніторингу й аналізу мереж: систем керування мережами, аналізаторів протоколів, мережних аналізаторів. Найпростішим варіантом експертної системи є контекстно-залежна help-система. Більш складні експертні системи являють собою так звані бази знань, що володіють елементами штучного інтелекту. Прикладом такої системи є експертна система, убудована в систему керування Spectrum компанії Cabletron.

У результаті проведення обстеження виходять наступні дані:

– перелік виявлених уразливостей (слабких місць) у настроюваннях устаткування, мережних сервісів, операційних систем, прикладного програмного забезпечення;

– докладний опис кожної виявленої уразливості, її розташування й оцінка можливих наслідків її використання зловмисниками;

– рекомендації з нейтралізації уразливостей (зниженню можливого збитку від їхнього використання зловмисниками), зміні конфігурації й настроювань компонент АС, використовуваних захисних механізмів, установці необхідних відновлень (patches, hot-fixes) установленого програмного забезпечення й т.п.

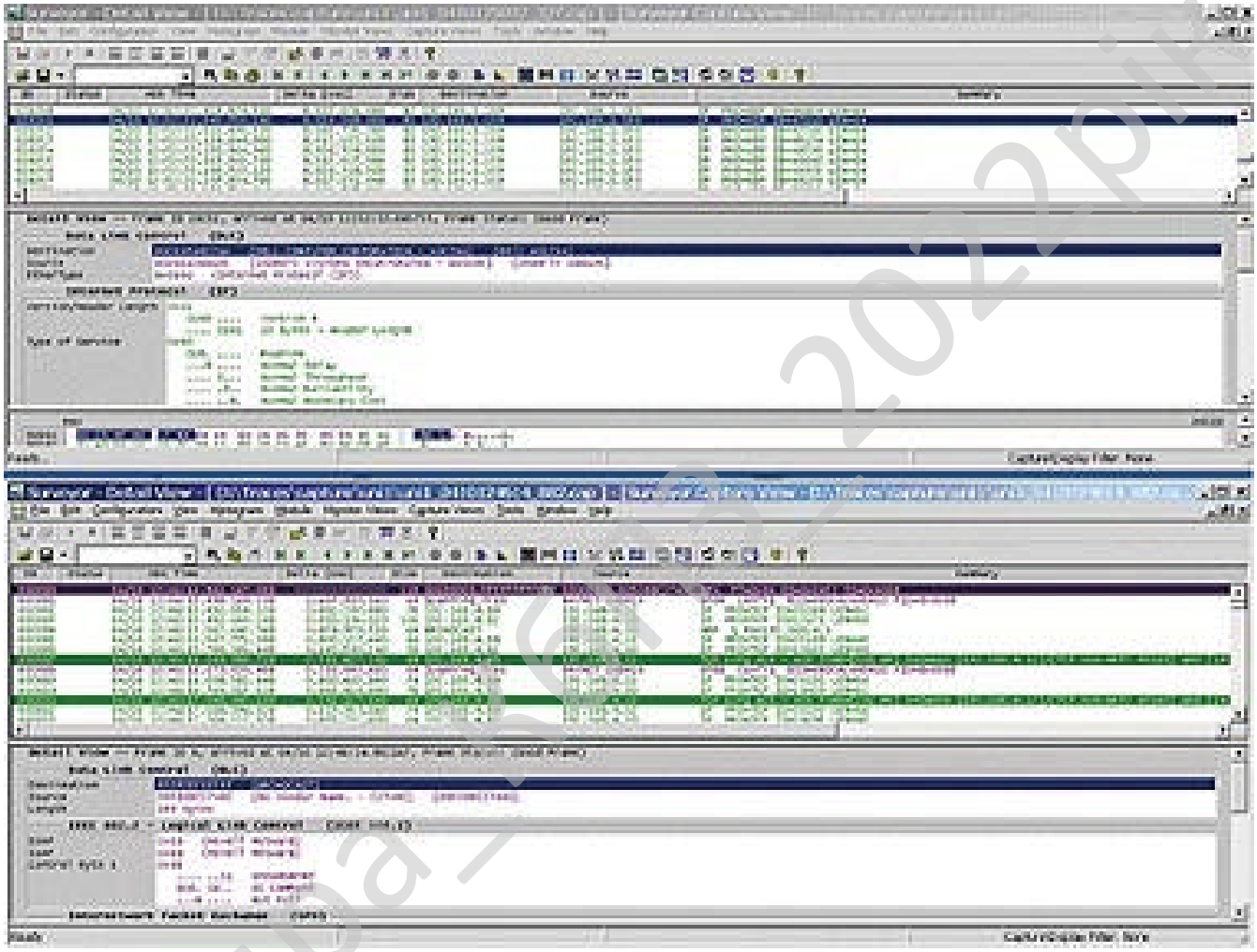


Рисунок 2.2 – Інтерфейс користувача Spectrum

Технічний аналіз мережі – це комплексна перевірка мережі, результатами якої є:

1. Письмовий висновок про якість роботи мережі:

– інформація про характеристики роботи мережі;

– дефектах;

– «вузьких місцях».

2. Рекомендації з поліпшення роботи мережі.

RiskWatch

Програмне забезпечення RiskWatch є потужним засобом аналізу й керування ризиками, більше орієнтованим на точну кількісну оцінку співвідношення втрат від погроз безпеки й витрат на створення системи захисту. Треба також відзначити, що в цьому продукті ризики в сфері інформаційної й фізичної безпеки комп'ютерної мережі підприємства розглядаються спільно. У сімейство RiskWatch входять наступні програмні продукти: для фізичних методів захисту, для інформаційних ризиків, для оцінки вимог до стандарту ISO 17799.

В основу продукту RiskWatch покладена методика аналізу ризиків, що складається із чотирьох етапів:

– перший – визначення предмета дослідження. Тут описуються такі параметри, як тип організації, состав досліджуваної системи, базові вимоги в області безпеки;

– другий – введення даних, що характеризують основні параметри системи. На цьому етапі докладно описуються ресурси, втрати й класи інцидентів. Останні виводяться шляхом зіставлення категорії втрат і категорії ресурсів. Крім того, задаються частота виникнення кожної з виділених погроз, ступінь уразливості й цінність ресурсів. Все це використовується надалі для розрахунку ефекту від впровадження засобів захисту;

– третій – кількісна оцінка. На цьому етапі розраховується профіль ризиків, і вибираються міри забезпечення безпеки. Фактично ризик оцінюється за допомогою математичного очікування втрат за рік. Ефект від впровадження засобів захисту кількісно описується за допомогою показника ROI (Return on Investment віддача від інвестицій), що показує віддачу від зроблених інвестицій за певний період часу;

– четвертий – генерація звітів.

Програмне забезпечення RiskWatch має масу достоїнств, а до недоліків продукту можна віднести його відносно високу вартість.

CRAMM

CRAMM – інструментальний засіб, що реалізує однойменну методику, що була розроблена компанією BIS Applied Systems Limited no замовленню британського уряду. Метод CRAMM дозволяє робити аналіз ризиків і вирішувати ряд інших аудиторських завдань: обстеження інформаційної системи, проведення аудита відповідно до вимог стандарту BS 7799, розробка політики безпеки.

Дана методика опирається на оцінки якісного характеру, одержувані від експертів, але на їхній базі будує вже кількісну оцінку. Метод є універсальним і підходить і для великих, і для дрібних організацій як урядового, так і комерційного сектора. Грамотне використання методу CRAMM дозволяє одержати дуже гарні результати, найбільш важливим з яких, мабуть, є можливість економічного забезпечення організації для забезпечення інформаційної безпеки безперервності бізнесу. Економічно обґрунтована стратегія керування ризиками дозволяє в остаточному підсумку уникати невиправданих витрат.

CRAMM припускає поділ всієї процедури на три послідовних етапи. Завданням першого етапу є визначення достатності для захисту системи застосування засобів базового рівня, що реалізують традиційні функції безпеки, або необхідність проведення більше детального аналізу. На другому етапі виробляється ідентифікація ризиків і оцінюється їхня величина. На третьому етапі вирішується питання про вибір адекватних контрзаходів. Для кожного етапу визначаються набір вихідних даних, послідовність заходів, анкети для проведення інтерв'ю, списки перевірки й набір звітних документів.

Достоїнства методу CRAMM: добре структурований і широко випробуваний метод аналізу ризиків; може використовуватися на всіх стадіях проведення аудиту безпеки інформаційних систем; в основі програмного продукту лежить об'ємна база знань по контрзаходах в області інформаційної безпеки, гнучкість і універсальність даного методу дозволяють його

використовувати для аудита інформаційної системи будь-якого рівня складності й призначення; даний метод дозволяє розробляти план безперервності бізнесу. До недоліків методу CRAMM можна віднести наступне: для його використання потрібен висококваліфікований аудитор; аудит по даному методі процес досить трудомісткий і може зажадати місяці безперервної роботи; генерує велику кількість паперової документації, що не завжди виявляється корисної на практиці; неможливо внести доповнення в базу знань CRAMM, що викличе певні труднощі при адаптації цього методу до потреб конкретної організації [3].

COBRA

Система COBRA є засобом аналізу ризиків і оцінки відповідності інформаційної системи стандарту ISO 17799. Дана система реалізує методи кількісної оцінки ризиків, а також інструменти для консалтингу й проведення оглядів безпеки. У систему COBRA закладений принцип побудови експертних систем, велика база знань по погрозах і уразливостях, велика кількість запитальників, з успіхом застосовуються на практиці.

КОНДОР+

Програмний продукт КОНДОР+ дозволяє фахівцям (IT-менеджерам, офіцерам безпеки) перевірити політикові інформаційної безпеки компанії на відповідність вимогам ISO 17799. КОНДОР+ містить у собі більше 200 питань, відповівши на які фахівець одержує докладний звіт про стан існуючої політики безпеки, а також модуль оцінки рівня ризиків відповідності вимогам ISO 17799. У звіті відбиваються всі положення політики безпеки, які відповідають і не відповідають стандарту, а також існуючий рівень ризику невиконання вимог політики безпеки у відповідності зі стандартом. Елементам, які не виконуються, даються коментарі й рекомендації експертів. За бажанням фахівця, що працює із програмою, можуть бути обрані генерація звіту, наприклад, по якомусь одному або декількох розділах стандарту ISO 17799, загальний докладний звіт з коментарями, загальний звіт про стан політики безпеки без коментарів для подання керівництву. Всі варіанти звітів для большеи наочності супроводжуються

діаграмами. КОНДОР+ дає можливість фахівцеві відслідковувати внесені на основі виданих рекомендацій зміни в політику безпеки, поступово приводячи її в повну відповідність із вимогами стандарту. Дана система реалізує метод якісної оцінки ризиків по рівневій шкалі ризиків: високий, середній, низький.

ГРИФ

ГРИФ – це програмний комплекс аналізу й контролю ризиків інформаційних систем компаній. У ньому розроблене гнучке й, незважаючи на схований від користувача дуже складний алгоритм, що враховує більше 100 параметрів, максимально простої у використанні програмне рішення, основне завдання якого дати можливість IT-менеджерові самостійно (без залучення сторонніх експертів) оцінити рівень ризиків в інформаційній системі й ефективність існуючої практики по забезпеченню безпеки компанії. Даний комплекс робить оцінку ризиків по різних інформаційних ресурсах, підраховує сумарний ризик по ресурсах компанії, а також веде підрахунок співвідношення збитку й ризику й видає недоліки існуючої політики безпеки. В основі продукту ГРИФ закладений методика аналізу ризиків, що складається з п'яти етапів:

– на першому визначається повний список інформаційних ресурсів, що представляють цінність у досліджуваній автоматизованій системі, які поєднуються в мережні групи;

– на другому здійснюється введення в систему всіх видів інформації, що представляє цінність для інформаційної системи. Введені групи коштовної інформації повинні бути розміщені користувачем на раніше зазначені на попередньому етапі об'єктах зберігання інформації (серверах, робочих станціях і т.д.); вказується збиток по кожній групі коштовної інформації, розташованої на відповідних ресурсах, по всіх видах погроз (конфіденційності, цілісності, відмови обслуговування);

– на третьому спочатку відбувається визначення всіх видів користувальницьких груп, потім визначається, до яких груп інформації на ресурсах має доступ кожна із груп користувачів. На закінчення визначаються

види (локальний і/або вилучений) і права (читання, запис, видалення) доступу користувачів до всіх ресурсів, що містять коштовну інформацію;

– на четвертому вказується, якими засобами захисту інформації захищені коштовна інформація на ресурсах і робітники місця груп користувачів. Уводиться інформація про витрати на придбання всіх засобів, що застосовуються, захисту інформації й щорічних витрат на їхню технічну підтримку, а також на супровід системи інформаційної безпеки;

– на завершальному етапі необхідно відповісти на список питань по політиці безпеки, реалізованої в системі, вирішує оцінити реальний рівень захищеності системи й деталізувати оцінки ризиків. Цей етап необхідний для одержання достовірних оцінок існуючих у системі ризиків.

Звітна система програмного комплексу ГРИФ складається із трьох частин: перша «Інформаційні ризики ресурсів», друга «Співвідношення збитку й ризику», третя «Загальний вивід про існуючі ризики інформаційної системи».

Авангард

Комплексна експертна система керування інформаційною безпекою «Авангард» є програмним продуктом, призначеним для рішення завдань керування безпекою в великих територіально-розподілених автоматизованих інформаційних системах, і покликана полегшувати завдання контролю за центральними структурами рівня забезпечення інформаційної безпеки на місцях. Даний комплекс є одним із самих потужних інструмент аналізу й контролю ризиків вітчизняного виробництва.

Основні можливості комплексу:

– гнучка система введення й редагування моделі підприємства;

– можливість побудови моделі ризиків;

– система оцінки й порівняння ризиків;

– оцінка мер протидії, побудова варіантів комплексів мер захисту й оцінка залишкового ризику.

Таким чином, програмний комплекс «Авангард» покликаний відігравати допоміжну роль у рішенні завдань керування інформаційної безпеки, а саме забезпечувати повноцінний всебічний аналіз, вирішує сформулювати обґрунтований набір цілей безпеки, обґрунтувати політикові безпеки, гарантувати повноту вимог безпеки, контроль виконання яких потрібно здійснювати [4].

Розглянуті методики дозволяють оцінити або переоцінити рівень поточного стану інформаційної безпеки автоматизованої системи, знизити потенційні втрати шляхом підвищення стійкості функціонування корпоративної мережі, розробити концепцію й політику безпеки автоматизованої системи, а також запропонувати плани захисту від виявлених погроз і уразливих місць. Важливо помітити, що на сьогоднішній день існують різноманітні й складні по своїй структурі автоматизовані системи, для яких неможливо підібрати конкретну методику оцінки ризиків, тому для одержання точних задовільних результатів оцінки необхідно використовувати комплексний підхід до оцінок ризиків на основі вже існуючих методик.

Таким чином, методику проведення аналізу ризиків підприємства й інструментальних засобів, що підтримують її, варто вибирати, з огляду на наступні фактори: наявність експертів в області оцінки ризику, статистики по інцидентах порушення інформаційної безпеки, точної кількісної оцінки або досить оцінки на якісному рівні.

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 формування профілів захисту хмарних сервісів.

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

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

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

Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розглянемо основні принципи, методи формування й структуру профілю захисту, а також характеристику стану інформаційних систем банківської сфери

Представимо огляд сучасних інформаційних банківських систем, їхню структура. Основні функціональні модулі систем, що реалізують всі види банківських послуг:

– розрахунково-касове обслуговування юридичних осіб;

– обслуговування рахунків банків-кореспондентів;

– кредитні, депозитні й валютні операції;

– будь-які види внесків приватних осіб і операції по них;

– фондові операції; розрахунки за допомогою пластикових карток;

– бухгалтерські функції;

– аналіз, прийняття рішень, менеджмент, маркетинг і ін.

Освітимо найбільш перспективні напрямки розвитку банківських інформаційних технологій, такі як:

– інтернет-банкінг;

– системи дистанційного обслуговування: «Інтернет-банк», «Інтернет-клієнт», домашній банк, телебанк, мобільний банк або WAP-сервіс.

З їхньою допомогою задовольняються практично будь-які, крім касового обслуговування, вимоги клієнтів банку. Освоєння українськими кредитними організаціями нових напрямків розвитку брокерських послуг полягає в наданні фізичним особам доступу до українських і міжнародних валютних і фондових ринків (інтернет-трейдінг).

Розглянемо стандарти в області інформаційної безпеки:

– міжнародні й національні стандарти оцінки керування інформаційною безпекою;

– галузеві стандарти забезпечення безпеки в банківській сфері;

– стандарти й рекомендації в області стандартизації:

а) забезпечення інформаційної безпеки організацій банківської системи України. Загальні положення СТО БР ІББС-1.0-2006;

б) методика оцінки відповідності інформаційної безпеки організацій банківської системи України вимогам СТО БР ІББС-1.0-2006;

в) посібник із самооцінки відповідності інформаційної безпеки організацій банківської системи України вимогам СТО БР ІББС-1.0-2006;

– аудит інформаційної безпеки.

Розглянемо стандарти й методичні рекомендації, присвячені формуванню й оцінці профілів захисту й завдань по безпеці відповідно до ДСТ ІСО/МЕК 15408. Представлено існуючі на даному етапі способи формування профілів захисту й завдань по безпеці. Особлива увага приділена складеним об'єктам оцінки (складається із двох і більше компонентів), якими і є в деяких випадках автоматизовані банківські системи. На рисунку 3.1 представлені два види складених об'єктів оцінки (ОО), з єдиним і різним середовищем безпеки.

Запропонуємо конкурентну модель СЗІ автоматизованих банківських систем комерційних банків і метод обґрунтування профілю захисту на основі даної моделі.

Введемо поняття конкуренції стосовно до інформаційних систем. Інваріантність даного поняття дає підставу припускати можливість побудови деякої математичної моделі даного процесу. Конкуренція, одержавши широке поширення в теорії еволюції біологічних і економічних систем, а також інших сферах, таких як політика, історія науки, утворення, мистецтво, соціальна психологія й навіть фізика, дозволяє використовувати дане поняття й у сфері інформаційних технологій. Зокрема, в області інформаційної безпеки.

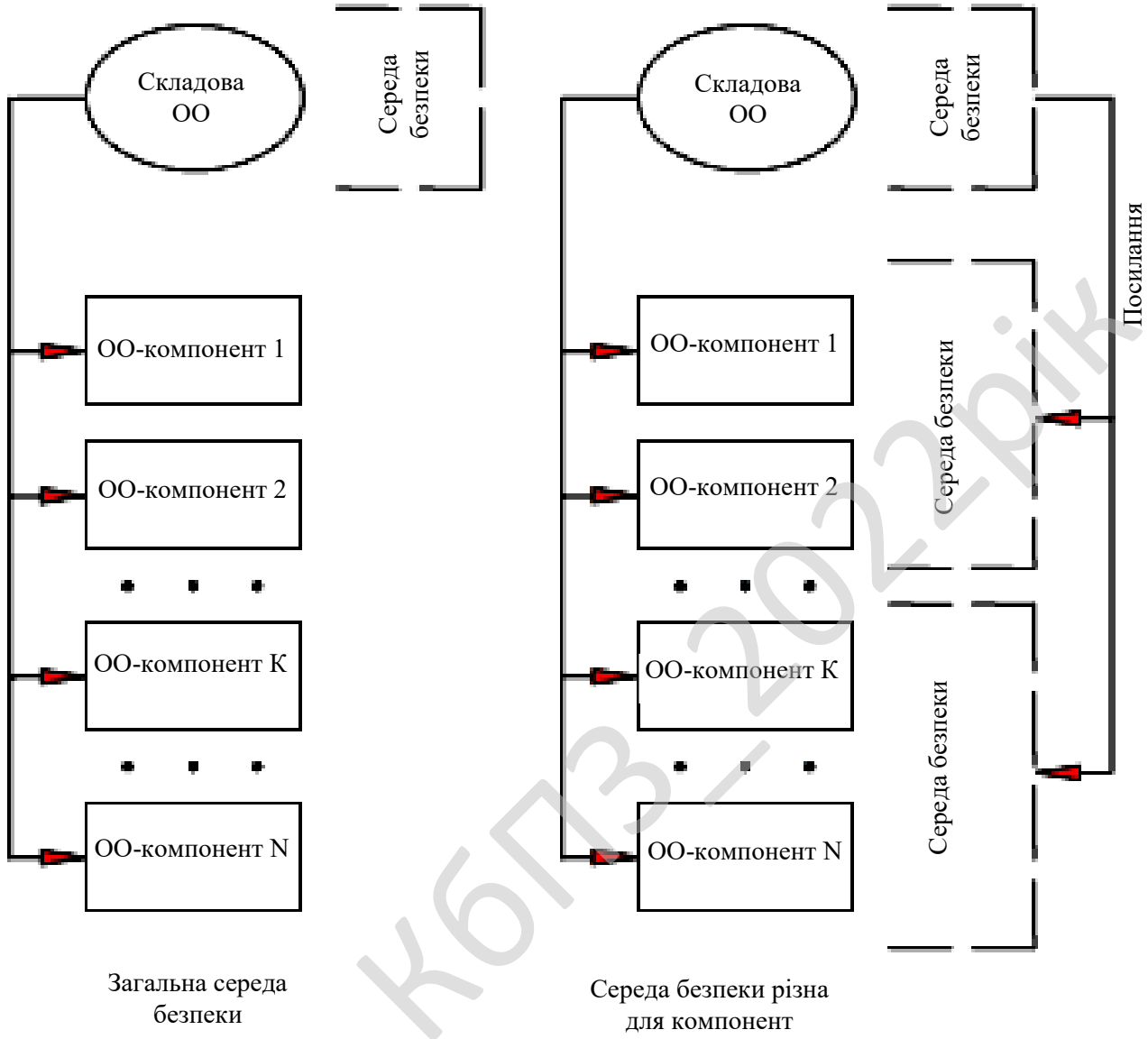


Рисунок 3.1 – Складені об'єкти оцінки ОО

Опишемо вимоги, запропоновані до моделі як до відбиття однієї зі сторін взаємодії, й не є повним функціональним аналогом реальної системи, а також вимоги до вибору економічного показника, що не повинен бути вузькоспеціалізованим параметром, характерним тільки для деяких систем, а повинен бути властивий всім учасникам взаємодії. Як такий показник запропонований використовувати чисті активи (нетто-активи), обумовлені як різниця між активами й пасивами:

$$А = АК – П, \quad (3.1)$$

де:

– $А$ – чисті активи (грн.);

– $АК$ – активи (грн.);

– $П$ – пасиви (грн.).

Для виключення впливу таких макроекономічних показників як середні доходи населення, середня заробітна плата, ціни, рівень інфляції, безробіття, зайнятість, продуктивність праці вводиться поняття нормалізованого активу:

$$AN_i = \frac{A_i}{\sum_{i=1}^{n} A_i}, \quad (3.2)$$

де:

– AN – нормалізований актив;

– $А$ – чистий актив (грн.);

– n – число діючих кредитних організацій.

У графічному виді представлена й проаналізований взаємозв'язок між чистими й нормалізованими активами п'яти найбільших банків.

Введемо визначення конкурентоспроможності як суми показника захищеності ($З$) і показника росту активів ($Н$):

$$КС = З + Н, \quad (3.3)$$

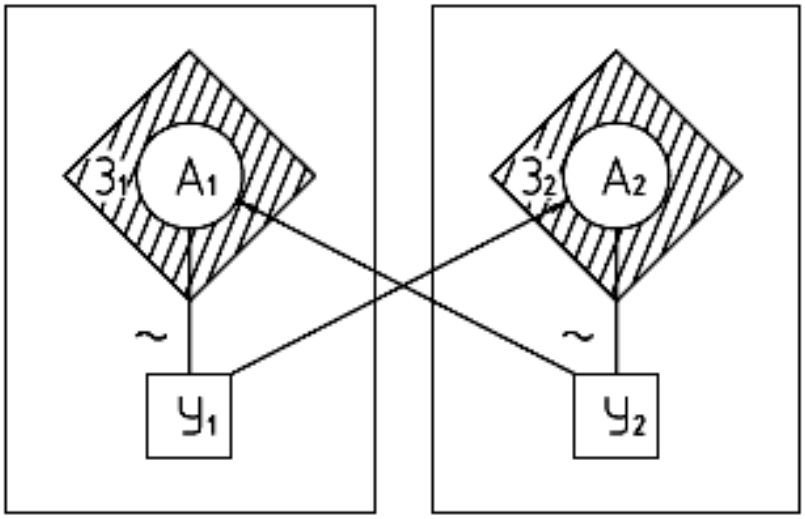
де:

– $КС$ – нормалізований актив;

– $З$ – захищеність;

– $Н$ – приріст активу.

Формулювання конкурентної моделі взаємодії систем СЗІ автоматизованих банківських систем комерційних банків представлені на рисунку 3.2.



Суб’єкт 1 Суб’єкт 2

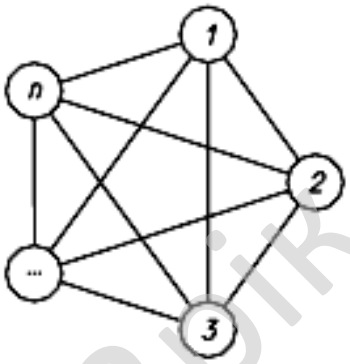


Рисунок 3.2 – Концептуальна схема конкурентної моделі СЗІ АБС КБ

Суб'єкт 1 має актив (A_1), що володіє тією або іншою захищеністю ($З_1$) від погроз, що виходять від другого (B_2). І, у той же час, сам формує погрозу (B_1), спрямовану на інший суб'єкт:

$$З_i(t) = \frac{\Delta AN_i(t)}{У_i}. \quad (3.4)$$

Захищеність суб'єкта ($З$) – це зміна нормалізованого активу AN від часу t-1 до часу t під впливом погрози (B):

$$\Delta AN_i(t) = AN_{i(t)} - AN_{i(t-1)}. \quad (3.5)$$

Погроза – імовірність втрати активу в одиницю часу. Як вихідне положення поле погроз для всіх суб'єктів системи єдино. Розходження погроз обумовлене як потужністю активу як з боку джерела, так і мети спрямованості погроз.

$$У_i = К_i \times \left(\sum_{i=1}^{n} К_i - К_i \right), \quad (3.6)$$

де:

– $У$ – погроза;

– $К$ – потужність активу.

Ріст своїх активів у результаті реалізації погроз, спрямованих на інші, у конкурентній моделі СЗІ автоматизованих банківських систем комерційних банків:

$$H = \frac{A_i^t - \sum_{i=1}^{n} A_i^t / \sum_{i=1}^{n} A_i^t \times A_i^{t-1}}{\sum_{i=1}^{n}\left(A_i^t - \sum_{i=1}^{n} A_i^t / \sum_{i=1}^{n} A_i^t \times A_i^{t-1} \right)}, \quad (3.7)$$

де:

H – приріст активу;

A – чистий актив (грн.).

У випадку, коли система не задовольняє вимозі конкурентоспроможності, за результатами експлуатації буде потрібно внесення розроблювачем виправлень в об'єкт оцінки, а також перевизначення вимог безпеки й/або припущень щодо середовища експлуатації, що спричинить перегляд профілю захисту.

Викладені вище положення не суперечать п.4.2.2 (Оцінка ОО) Державного стандарту України ДСТ ІСО/МЕК 15408-1-2002, у якому регламентоване, що процес оцінки може проводитися як паралельно з розробкою, так і слідом за нею.

Опишемо метод виконання оцінки на базі моделюючого комплексу. Метод містить у собі наступні етапи:

1. Одержання вихідних даних і прийняття ряду положень.

2. Розрахунок необхідних показників (нормалізований актив, захищеність, потужність активу, зміна активу за рахунок перерозподілу, конкурентоспроможність).

3. Ухвалення рішення про відповідність профілю захисту.

1-й етап: Збір статистичних даних, отриманих з відкритих джерел, за результатами діяльності банків і кредитних організацій.

Достатньою умовою є одержання даних по 500 самих великих організаціях. Вплив інших незначний, дані по них екстраполюються. Одержання свідчення про відповідність профілю захисту конкретної реалізації системи.

2-й етап: Обчислення нормалізованого активу за формулою (3.2): n – приймаємо рівним 1125, а потім і на часовому інтервалі від t-1 до t (3.5). Величина інтервалу приймається залежно від даних, отриманих на 1-му етапі (0,5 року). Визначаємо потужність активу кожного суб'єкта відповідно до його активу.

Ріст активів у результаті реалізації погроз, спрямованих на інші, являє собою відхилення від середнього приросту активів протягом часу від від t-1 до t. Обчислюється за формулою (3.7).

Обчислення захищеності виробляється за формулою (3.4), де B приймається за формулою (3.6). Таким чином, величина погрози B являє собою добуток потужності власного активу на суму потужностей активів інших організацій. Безпосереднє значення погрози, як імовірності втрати активу, не враховується, тому що є константою для всіх суб'єктів. При обчисленні значення захищеності вхідними параметрами виступають величина суб'єкта і його нормалізований актив.

Останнім значенням обчислюється параметр КС (конкурентоспроможність). Результатом є сума показників захищеності й показника росту активу.

3-й етап: Ухвалення рішення про відповідність профілю захисту на підставі показника КС. Для успішного розвитку в майбутньому показник КС повинен приймати, як мінімум, значення вище за середнє, тому що захищеність має тенденцію до зниження протягом часу, а разом з нею знизиться й показник КС .

3.2 Розробка структурної схеми

Структурна схема розробленої системи зображена на рисунку 3.3. В основному інтелектуальні засоби захисту інформації (СЗІ) знайшли своє застосування в системах виявлення атак як інтелектуальний інструмент, у яких,

як правило, використовуються нейронні мережі (НМ), системи нечіткої логіки (НЛ) і засновані на правилах експертні системи (ЕС) [1].

Схеми виявлення атак розділяють на дві категорії:

1) виявлення зловживань;

2) виявлення аномалій.

До першої відносять атаки, які використовують відомі уразливості інформаційної системи (ІС), а до других – невластиву користувачам ІС діяльність.

Для виявлення аномалій виявляється діяльність, що відрізняється від шаблонів, установлених для користувачів або груп користувачів. Виявлення аномалій, як правило, пов'язане зі створенням бази знань (БЗ), що містить профілі контрольованої діяльності [2], а виявлення зловживань – з порівнянням діяльності користувача з відомими шаблонами поводження хакера [3] і використовує методи на основі правил, що описують сценарії атак. Механізм виявлення ідентифікує потенційні атаки у випадку, якщо дії користувача не збігаються із установленими правилами.

Завдання класифікації в експертних системах

Експертні системи (рисунок 3.3) призначені для рішення класифікаційних завдань у вузькій предметній області виходячи з бази знань, сформованої шляхом опитування кваліфікованих фахівців і представленою системою класифікаційних правил *If-Then* (*Якщо – Тоді*) [4]. У системах забезпечення безпеки ІС експертні системи використовуються в інтелектуальних СЗІ на основі моделі [5] і містять у БЗ опис класифікаційних правил, що відповідають профілям легальних користувачів ІС, сценаріям атак на ІС [6].

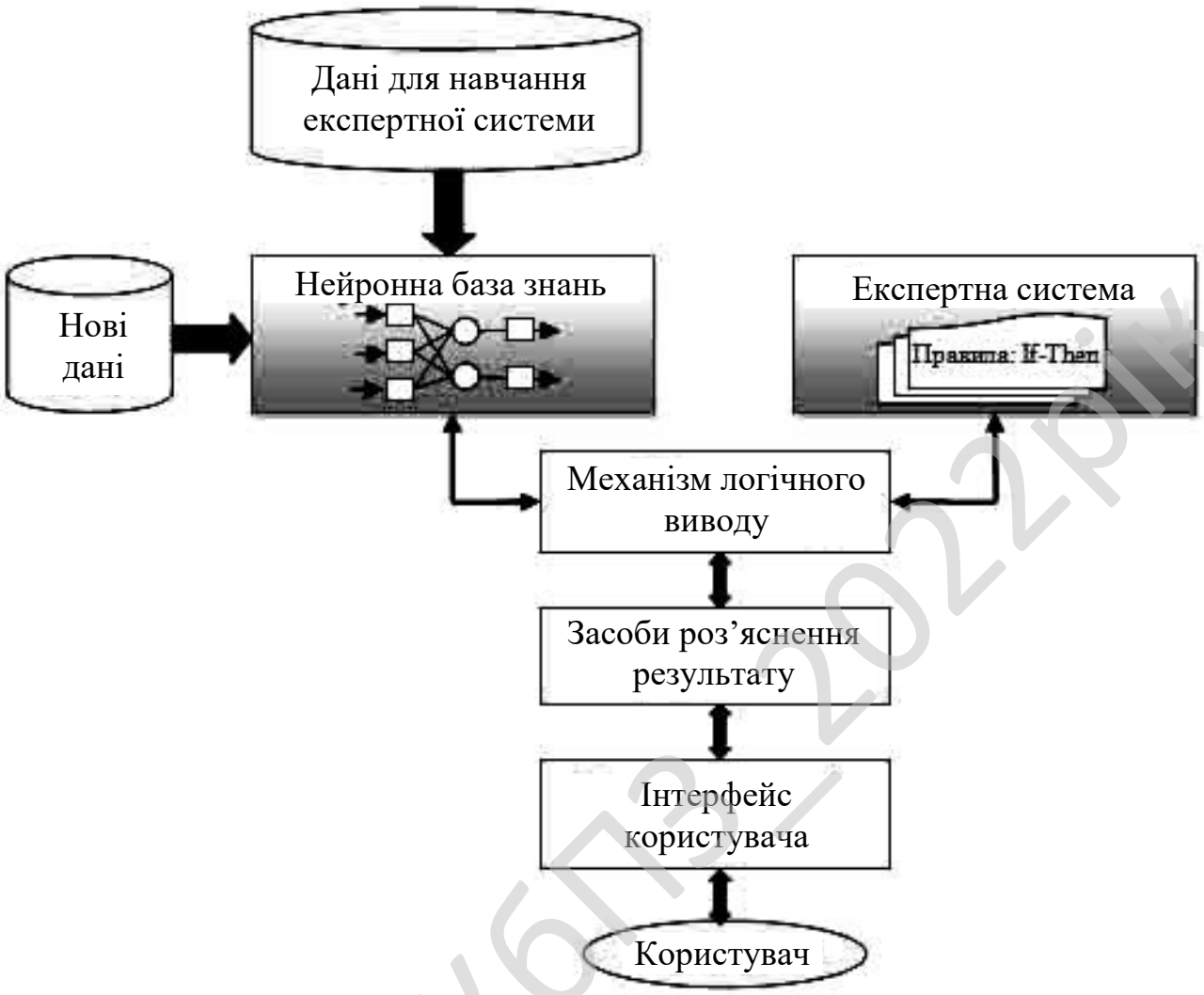


Рисунок 3.3 – Структурна схема системи

До недоліків ЕС, як засобів класифікації, відносять [7]:

– Непрозорість зв'язків між окремими правилами в базі знань. Хоча окремі правила відносно прості й логічно прозорі, наочність їхнього логічного взаємозв'язку в межах БЗ має бути досить низькою, тобто не просто визначити суперечливі правила в БЗ і їхню роль у рішенні завдання.

– Неефективна стратегія пошуку. ЕС із великою базою знань можуть виявитися недостатньо продуктивними для рішення оперативних завдань забезпечення безпеки ІС у реальному масштабі часу.

– Відсутність можливості адаптації. ЕС не мають здатність до автоматичного навчання, тобто ЕС не може автоматично змінювати БЗ, коректувати існуючі правила або додавати нові правила *If-Then.*

Імовірнісні методи рішення завдання класифікації

Методи недостовірного керування й імовірнісних міркувань застосовуються в ЕС не тільки для формулювання класифікаційних висновків відповідно до правил *If-Then,* але й формування оцінок вірогідності проведеної класифікації у вигляді значень значення фактора впевненості або умовної ймовірності виникнення класифікуємої події.

Можливість оцінки вірогідності пророкування є достоїнством методу імовірнісних міркувань Байеса, тому що базується на математичному апараті теорії ймовірностей [4]. У вітчизняній практиці метод імовірнісних міркувань Байеса застосовують у ряді прикладних ЕС [8].

Практична значимість методу факторів упевненості для рішення завдання класифікації підтверджена розробкою ряду експертних систем [9]. Останній підхід до класифікації більше прийнятний з погляду обчислювальної ефективності, тому що не вимагає наявності більших обсягів статистичних даних і складних розрахунків умовних ймовірностей при великій розмірності простору вхідних посилок [7].

У всіх додатках забезпечення безпеки ІС на основі експертних систем може успішно застосовуватися підхід формування й підтримки класифікаційної бази знань відповідно до методів недостовірного керування й імовірнісних міркувань.

Чисельна оцінка класифікаційних висновків особливо важлива в умовах неповноти й низкою вірогідності вхідних ознак, використовуваних як посилки більшістю існуючих систем класифікації вторгнень у комп'ютерні системи.

Завдання нечіткої класифікації

Нечітка класифікація є подальшим розвитком підходу до рішення експертними системами завдань класифікації. Достоїнство нечіткої класифікації – можливість формулювати достовірні класифікаційні висновки виходячи з неповних і не цілком достовірних вхідних посилок [7].

При збереженні математичного апарата, розробленого для систем чіткої

логіки, у нечітких логічних системах вирішене завдання перетворення чисельної і якісної інформації в ступінь приналежності значень конкретним нечітким безлічам [10]. Нечіткі безлічі описуються за допомогою функцій приналежності, що ставлять у відповідність безлічі значень із області визначення безперервної змінної безліч значень істинності з інтервалу [0, 1].

Етапи нечіткого логічного виводу безпосередньо пов'язані із процесом формування класифікаційних висновків [11]:

1) етап введення нечіткості (fuzzification) пов'язаний з перетворенням по засобом вхідних функцій приналежності (input membership functions) кожного із чітких вхідних значень x_i , $i = 1,\ldots,n,$ де n – число вхідних значень класифікатора (crisp inputs) у ступінь істинності відповідної посилки μ_{xi}, $i= 1,\ldots,m,$ де m-для кожного із класифікаційних правил (fuzzy rules);

2) етап нечіткого логічного виводу відповідає формуванню висновку (відповідні нечіткі підмножини) по кожному із правил μ_{Ri}, $i = 1,\ldots,m,$ де m – кількість класифікаційних правил, виходячи зі ступеня істинності посилок μ_{xi}, $i= 1,\ldots,n$;

3) етап композиції нечітких підмножин по кожному із правил μ_{Ri}, $i = 1,\ldots,m$ за допомогою вихідних функцій приналежності (output membership functions) з метою формування нечітких підмножин класифікаційних висновків μ_{Ci}, $i = 1,\ldots,p,$ де p – число виходів класифікатора;

4) етап об'єднання (aggregation) нечітких підмножин μ_{Ci}, $i = 1,\ldots,p$ і приведення до чіткості (defuzzification) приводить до формування вихідного чіткого значення *в.*

Нечіткі логічні системи зберігають у своєму составі базу знань кваліфікованих фахівців ІБ у вигляді системи правил *If-Then,* однак розширюють область застосування ЕС за рахунок рішення завдання класифікації виходячи з неповної й не цілком достовірної інформації.

Системи НЛ мають обмежені можливості до адаптації, тому що можуть навчатися шляхом зміни параметрів функцій приналежності під реальні значення

вхідних даних і бажаних класифікаційних висновків. Варто відзначити, що процес навчання функцій приналежності нечіткої ЕС із досить великою базою знань (понад 100 правил) трудомісткий і вимагає значних витрат часу [7].

Застосування НМ у завданнях класифікації й кластеризації

Нейронні мережі найбільше часто використовують для рішення завдань класифікації. Доведено, що НМ є універсальним аппроксіматором, тобто будь-яка функція може бути представлена у вигляді багатошарової НМ із формальних нейронів з нелінійною функцією активації. Формально підтверджена верхня границя складності НМ, що реалізує довільну безперервну функцію від декількох аргументів. Нейронною мережею з одним схованим шаром і прямими повними зв'язками можна представити будь-яку безперервну функцію, для чого досить у випадку n-мірного вхідного вектора $2n+1$ ФН схованого шару із заздалегідь застереженими обмеженими функціями активації [10].

Відомі численні застосування нейромережних засобів для забезпечення безпеки ІС, причому більшість випадків пов'язане з рішенням завдань класифікації й кластеризації [10].

Варто врахувати, що із всіх розглянутих раніше інтелектуальних засобів тільки НМ обдають властивістю самоорганізації, вирішує використовувати їх для рішення завдання кластеризації.

Можливість самоорганізації розглядається як одне з найбільш важливих якостей нейромережних СЗІ, вирішує адаптуватися до зміни вхідної інформації. Навчальним фактором виступають присутні в даних сховані закономірності й надмірність вхідної інформації. Інформаційна надмірність дозволяє фіксувати в інформаційному полі НМ вхідні дані, представляючи їх у більше компактній формі. Зменшення ступеня надмірності інформації в адаптивних СЗІ дозволяє виділяти істотні незалежні ознаки в даних.

Самоорганізація НМ реалізується за рахунок механізму кластеризації: подібні вхідні дані групуються нейронною мережею відповідно до взаємної кореляції й представляються конкретним ФН-прототипом. НМ, здійснюючи

кластеризацію нечітких даних, знаходить такі усереднені по кластеру значення ваг ФН-прототипів, які мінімізують помилку подання згрупованих у кластер даних.

Розглянуті механізми класифікації й кластеризації вхідних даних у СЗІ дозволяють не тільки відносити класифікуємий об'єкт (вектор вхідних даних) до одного з відомих класів, але й реалізувати еволюційні процеси самоорганізації, адаптації, розвитку в інтелектуальних засобах забезпечення інформаційної безпеки ІС. Причому кращі функціональні характеристики виходять при сполученні різних інтелектуальних засобів у гібридній системі захисту інформації [12].

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.4. Життєвий цикл процесу ухвалення рішення по формуванню профілю захисту в інформаційній системі (ІС), починається з аналізу проблемної ситуації особами, відповідальними за ініціацію процесу ухвалення рішення. Перш ніж ухвалити рішення щодо шляхів вирішення проблемної ситуації, відповідальні особи повинні зрозуміти зміст і ступінь її актуальності, і цей зміст осягається не індивідуально, а в процесі колективного всебічного аналізу проблемної ситуації, що здійснюється на 1 етапі життєвого циклу процесу ухвалення рішення по формуванню профілю захисту в ІС.

На даному етапі ініціатор процесу ухвалення рішення по формуванню профілю захисту в ІС, маючи дані про виниклу проблемну ситуацію, готовить первинний її опис і обґрунтовану пропозицію по її вирішенню, після чого передає обґрунтовану пропозицію з вирішення проблемної ситуації в орган, уповноважений за організацію процесу ухвалення рішення по формуванню профілю захисту в ІС. Після одержання обґрунтованої пропозиції з вирішення виниклої проблемної ситуації орган, уповноважений за організацію процесу

ухвалення рішення, повинен організувати всебічний його аналіз із залученням експертів, після чого прийняти рішення, чи заслуговує виникла проблемна ситуація вирішення, чи ні. Для цього створюється експертна група, що оцінює актуальність проблемної ситуації, і проводить її експертизу, метою якої є максимально наочний опис виниклої проблеми в термінах предметної області. Після підготовки експертного висновку, де визначена актуальність проблемної ситуації й намічені основні шляхи її рішення, орган, уповноважений за організацію процесу ухвалення рішення по формуванню профілю захисту в ІС, ухвалює рішення щодо включенні проблемної ситуації до реєстру прийнятих рішень, що у свою чергу запустить наступний етап процесу ухвалення рішення . Орган, уповноважений за організацію процесу ухвалення рішення по формуванню профілю захисту в ІС, на основі експертного висновку може також відхилити пропозиція з вирішення проблемної ситуації, якщо вона визнана експертами несуттєвою або недостатньо актуальною.

На другому етапі процесу прийняття рішень – етапі постановки завдання – здійснюється вибір найбільш прийнятного варіанта вирішення завдання (будь-яка складна проблемна ситуація припускає кілька можливих шляхів виходу зі сформованої ситуації, серед яких найчастіше буває дуже складно вибрати найбільш ефективний), а також підготовка й узгодження правового акту, вирішує проблемну ситуацію.

На етапі ухвалення рішення (третій етап) підготовлений правовий акт, вирішує проблемну ситуацію, розглядається на засіданні колегіального органа ухвалення рішення або керівництвом організації. Після затвердження даний правовий акт включається до реєстру правових актів.

Нарешті, на вирішальному четвертому етапі процесу ухвалення рішення по формуванню профілю захисту в ІС здійснюється моніторинг і оцінка результативності виконання ухваленого рішення органами виконавчої влади, результатом якого є експертний висновок з оцінкою результативності ухваленого рішення. На основі даного експертного висновку приймається рішення про

включення виконаного рішення до реєстру прецедентів проблемних ситуацій. Це необхідно для того, щоб забезпечити можливість нагромадження успішного досвіду дозволу проблемних ситуацій з метою його наступного використання в майбутньому при розробці рішень при виникненні аналогічних проблем.

Важливо відзначити, що процес прийняття управлінських рішень, по своїй природі, носить ітераційний характер. Це означає, що якщо отриманий варіант рішення проблемної ситуації за якимись причинами не влаштовує відповідальних осіб, здійснюється повернення на попередні етапи, де виробляється новий варіант рішення проблемної ситуації. Такі ітерації тривають доти, поки результат не буде погоджений всіма відповідальними особам або, можливо, ситуація зміниться й відпаде необхідність у вирішенню проблемної ситуації.

Аналіз періодично виникаючих проблемних ситуацій в області інформаційної безпеки, показує, що причина їхнього виникнення, насамперед, полягає у відсутності єдиної системи обліку однотипних рішень по одній проблемній ситуації з різних джерел.

Для усунення подібної ситуації було ухвалено рішення про вдосконалювання існуючої централізованої автоматизованої інформаційної системи забезпечення інформаційної безпеки. Поставлено завдання:

1. Створити єдину систему, що буде узагальнювати за схемою: проблема – проблемна ситуація – проблемний напрямок всі проблемні питання, рішення яких повинен забезпечити відділ захисту інформації.

2. На підставі єдиного реєстру проблемних ситуацій організувати більш ефективний часовий і змістовний моніторинг ефективності їхнього рішення.

Управлінським ефектом від реалізації 1 і 2 пунктів стане формування інструмента, що реально стимулює взаємодію між відділами підприємства.

Практична цінність від впровадження подібного програмного забезпечення полягає у формуванні єдиної системи обліку проблемних питань пов'язаних із захистом інформації на підприємстві, поставлених для рішення керівництвом організації. Наявність даного переліку не дозволить (при всьому

бажанні) проігнорувати проблемну ситуацію відповідальною особою. Реально виникає механізм, який дозволяє інструментально вирішити проблему усунення суб'єктивних бар'єрів взаємодії між відділами.

Реалізація викладеного підходу до прийняття міжгалузевих колегіальних рішень практично неможлива без використання спеціальних програмних засобів. На рисунку 3.4 представлена архітектура інформаційної системи для підтримки прийняття рішень по формуванню профілю захисту в інформаційній системі.

Функціональна архітектура інформаційної системи підтримки процесу прийняття рішень реалізована у відповідності с розглянутою моделлю процесу ухвалення рішення по формуванню профілю захисту в ІС.

Онтоло́гія – це спроба всеосяжної й детальної формалізації деякої області знань за допомогою концептуальної схеми. Звичайно така схема складається зі структури даних, утримуючі всі релевантні класи об'єктів, їхні зв'язки й правила (теореми, обмеження), прийняті в цій області. Цей термін в інформатиці є похідним від древнього філософського поняття «онтологія». В області штучного інтелекту онтологією називається експліцитна специфікація концептуалізації. Онтології використовуються в процесі програмування як форма подання знань про реальний світ або його частину. Основні сфери застосування – моделювання бізнес-процесів, семантична павутина (Semantic Web), штучний інтелект.

Склад модулів інформаційної системи підтримки прийняття рішень повною мірою відповідає завданням забезпечення необхідної функціональності системи формування профілю захисту в інформаційній системі, у тому числі підтримує завдання:

– забезпечення доступу співробітників підприємства, експертів і фахівців підвідомчих організацій до відповідних інформаційних ресурсів (баз знань і баз даних);

– забезпечення керування електронними процедурами в процесі прийняття й узгодження рішень по формуванню профілю захисту в інформаційній системі;

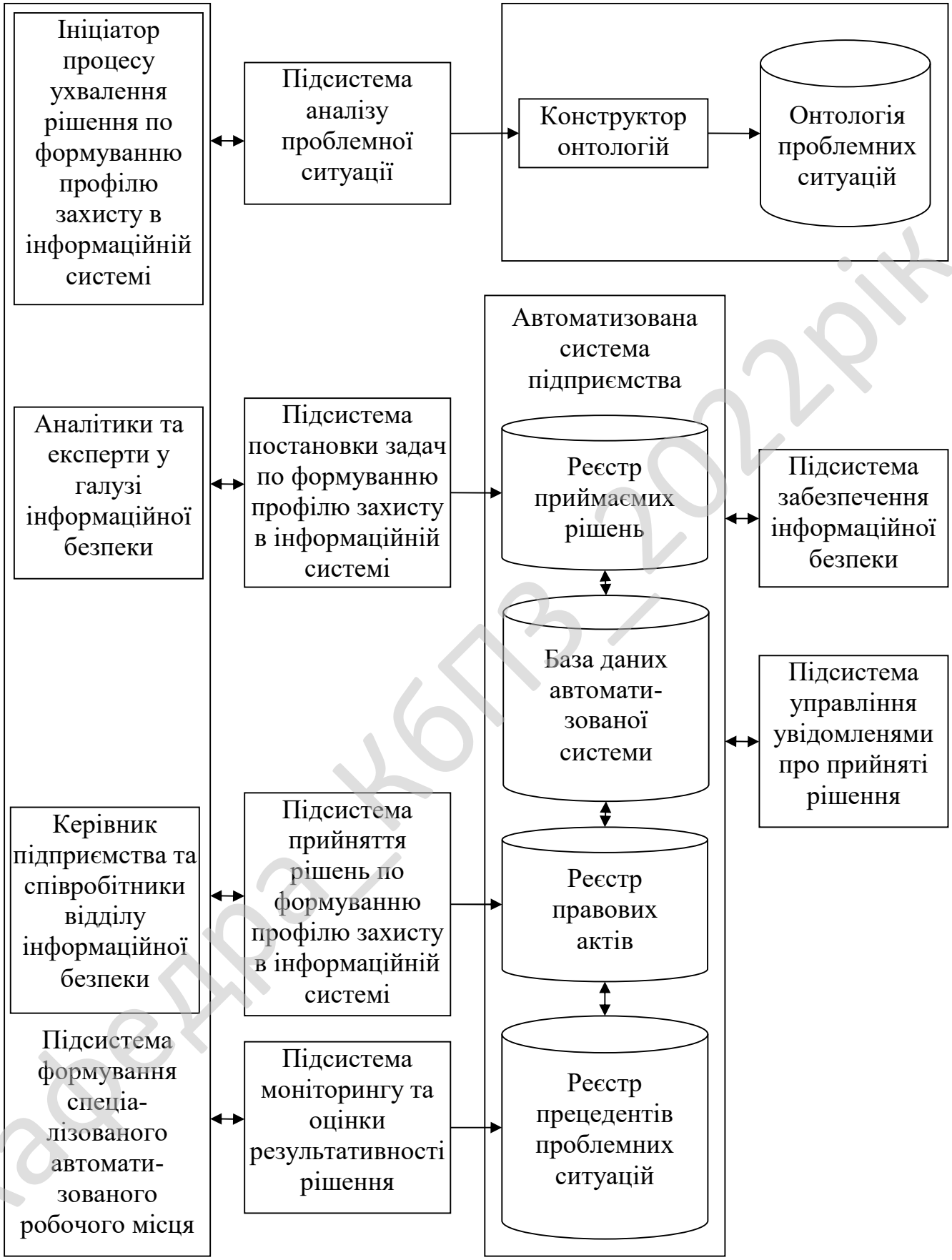


Рисунок 3.4 – Функціональна схема системи

– забезпечення в організації інформаційно-управлінської взаємодії усередині підприємства;

– забезпечення формування інформаційної інфраструктури, що підтримує експертні взаємодії, у тому числі організацію експертної групи, підтримку організаційних і інформаційних взаємодій усередині групи, обмін експертною інформацією в процесі прийняття управлінських рішень;

– забезпечення збору, обробки й зберігання експертної інформації (інформації про проблемні ситуації), а також застосування до неї аналітичних інструментів;

– забезпечення формування електронних архівів: реєстру прийнятих рішень, реєстру правових актів, реєстру прецедентів проблемних ситуацій. При цьому доступ до даних реєстрів, у рамках прав доступу, забезпечується в рамках автоматизованої системи підприємства

– забезпечення оперативного інформування про хід процесу ухвалення управлінського рішення;

– забезпечення прозорості діяльності учасників процесу прийняття управлінських рішень;

– забезпечення моніторингу й контролю виконання прийнятих рішень;

– забезпечення пошуку в інформаційних ресурсах системи;

– забезпечення захисту інформаційних ресурсів і програмно-технологічної інфраструктури від погроз інформаційної безпеки;

– забезпечення керування правами доступу до інформації й повноваженнями по роботі в системі, включаючи повноваження по зміні структури організації інформаційної системи (організації інформаційних взаємодій).

Основними результатами, очікуваними від впровадження й використання пропонованих методів і програмних засобів ситуаційного керування в регіоні, є наступні:

– Формування єдиної системи обліку проблемних ситуацій по інформаційній безпеці.

– Відстеження в реальному часі розвиток проблемної ситуації.

– Формування повної історії рішення проблемної ситуації від її виникнення до оцінки результатів, створення бібліотеки колективного досвіду знаходження виходу з однотипних проблемних ситуацій.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.5. спершу завантажується процес запуску головного вікна програми. Він взаємодіє з наступними процесами:

– Процес моніторингу та оцінки результативності рішення.

– Процес прийняття рішень по формуванню профілю захисту.

– Процес постановки задач по формуванню профілю захисту.

– Процес аналізу проблемної ситуації.

Процес моніторингу та оцінки результативності рішення взаємодіє з процесом реєстру прецедентів проблемних ситуацій.

Процес прийняття рішень по формуванню профілю захисту взаємодіє з процесом реєстру правових актів.

Процес постановки задач по формуванню профілю захисту взаємодіє з наступними процесами:

– Процес реєстру приймаємих рішень.

– Процес роботи бази даних автоматизованої системи.

Процес аналізу проблемної ситуації взаємодіє з процесом конструктора онтологій, який у свою чергу взаємодіє з процесом онтології проблемних ситуацій. Онтологія – представлення деякою мовою знань про певну предметну область. Онтологію неодмінно супроводжує деяка концепція цієї області інтересів. Найчастіше ця концепція виражається за допомогою визначення

базових об’єктів (атрибутів, процесів) і відношень між ними. Визначення цих об’єктів і відношень між ними зазвичай називають концептуалізацією. Наступне визначення онтології є узагальнюючим: Онтологія – це загальноприйнята і загальнодоступна концептуалізація певної області знань (середовища), яка містить базис для моделювання цієї області знань і визначає протоколи для взаємодії між агентами, які використовують знання з цієї області, і, нарешті, включає домовленості про представлення теоретичних основ даної області знань.

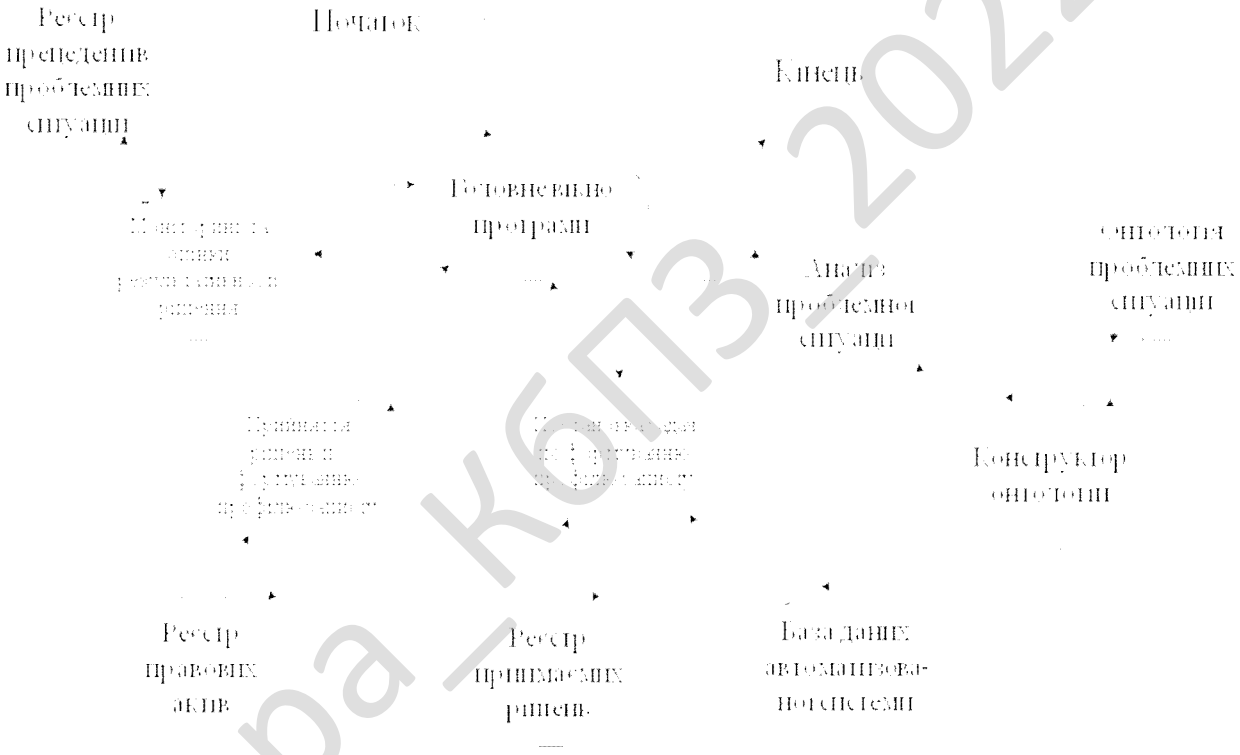


Рисунок 3.5 – Діаграма взаємодії процесів

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.

Якщо система була взламана, тоді відбувається обчислення показників стійкості застосованого алгоритму шифрування.

Наступним кроком є виведення на екран значень показників стійкості.

Й останнім кроком роботи системи є виведення на екран рекомендацій по забезпеченню стійкості системи.

Після цього користувач може вийти з програми, або запустити обробку іншого файлу системи.

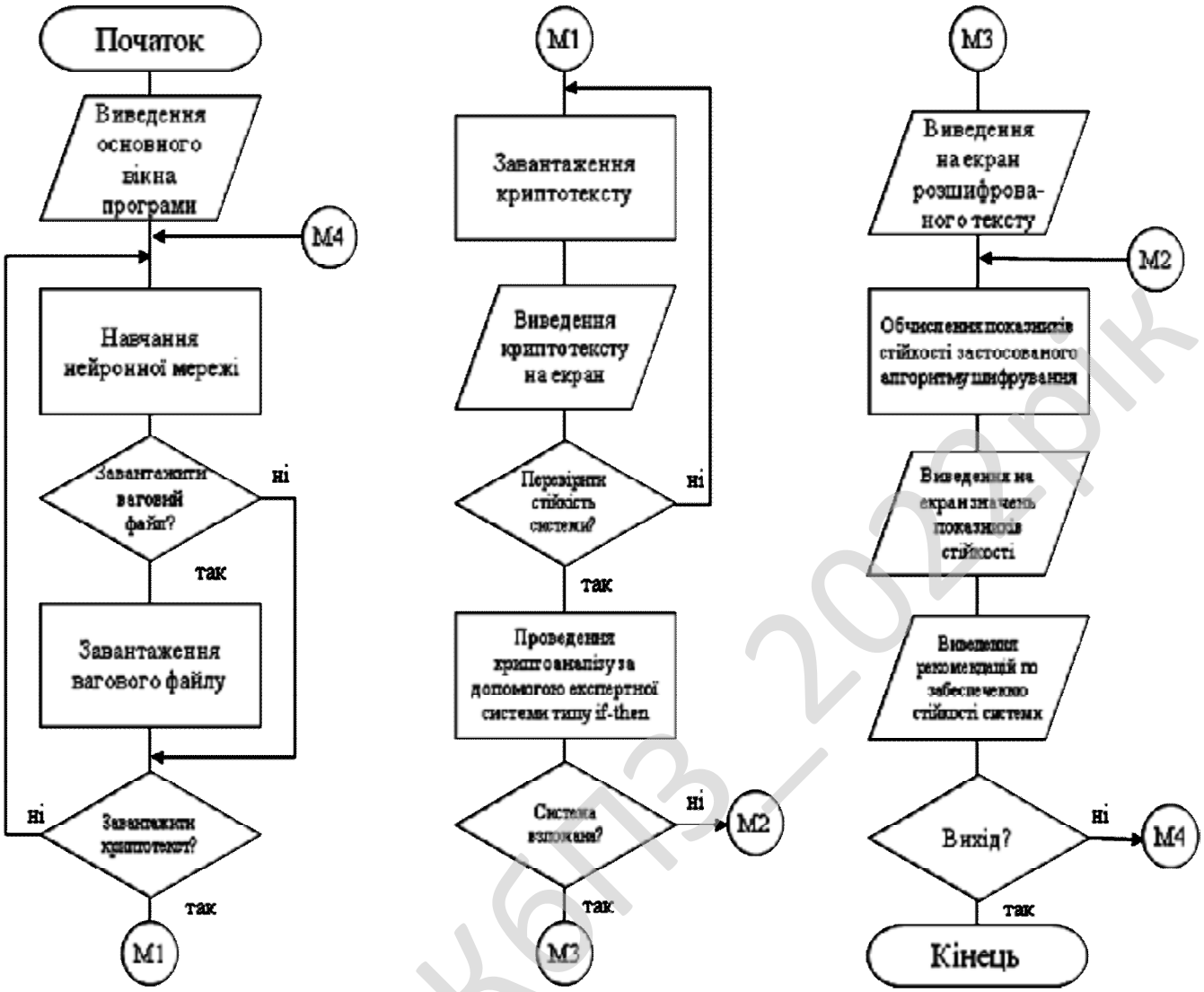


Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

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.

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.

Нейронні мережі можуть навчатися 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 $O(\sqrt{\#M})$.

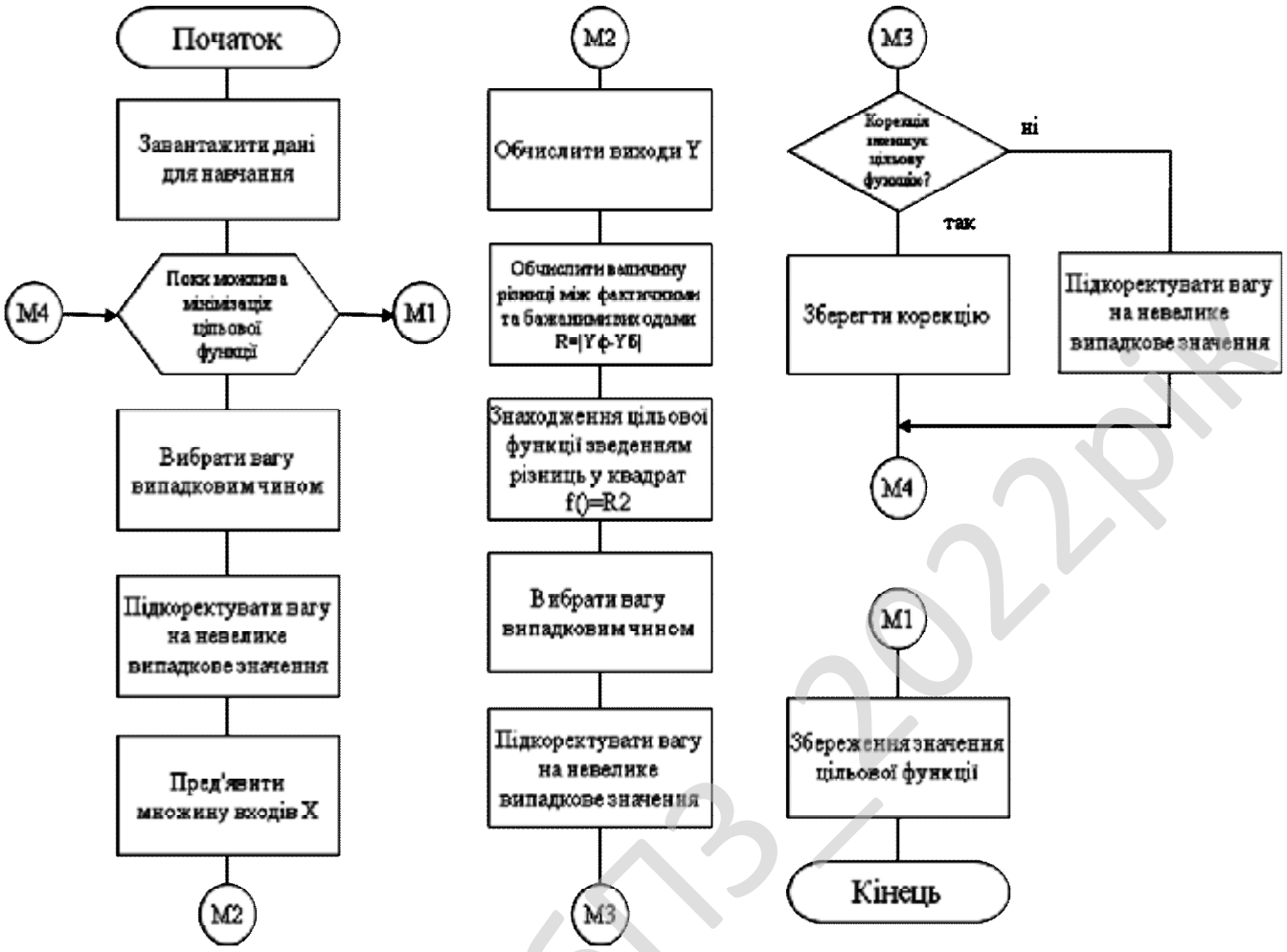


Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми навчання нейронної мережі

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 хеш-функції) для обчислювальної моделі з обсягом пам'яті $O(v)$. Такими аргументами будуть елементи множини M, стрілки від яких під дією хеш-функції f ведуть у точку входу в цикл. Практично алгоритм зводиться до знаходження точки входу в цикл.

Алгоритм роботи підпрограми аналізу стійкості криптосистеми (рисунок 4.3):

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.

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 $O(\sqrt{\#M}\log_v \#M)$.

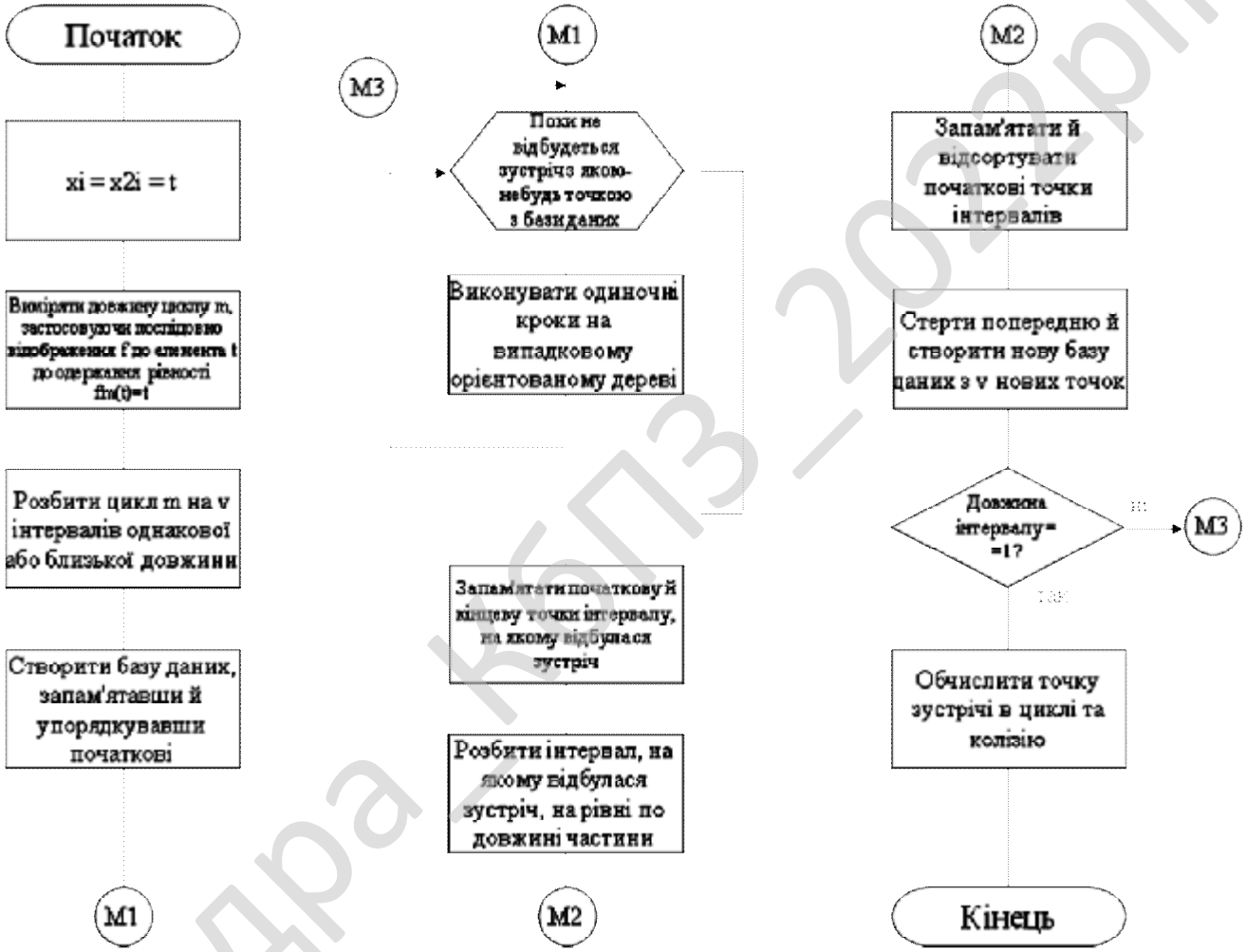


Рисунок 4.3 – Блок-схема алгоритму роботи підпрограми аналізу стійкості криптосистеми

Структура розробленої експертної системи

Проект «Expert»

Проект «Expert» служить для створення бази знань про погрози інформаційній безпеці й методи протидії даним погрозам, а також для редагування вже існуючої бази знань. Модуль «Unit1» проекту служить показовою формою при завантаженні експертної системи в режимі придбання знань. У цьому режимі експерт, використовуючи компонент придбання знань, наповнює систему знаннями, які дозволяють ЕС у режимі рішення самостійно (без експерта) вирішувати завдання із проблемної області. Експерт описує проблемну область у вигляді сукупності даних і правил. У цьому модулі експерт безпосередньо створює саму базу знань, формує правила. При цьому є можливість редагування правил, видалення правил, збереження бази знань, завантаження раніше збереженої бази знань. Модуль «Unit1» і модуль «Unit 2» необхідні для введення експертом питань до змінного умови й рекомендацій до змінних виводу відповідно.

Модуль «Unit4» призначений для виводу довідки про автора.

У програмі використовуються процедури й функції, які написані для зручності роботи й розуміння програми:

– procedure dobav_uslclick(sender: tobject) – додавання умови в правило;

– procedure formcreate(sender: tobject) – початкові установки при відкритті форми;

– procedure udal_uslclick(sender: tobject) – видалення умови в правилі;

– procedure newclick(sender: tobject) – формування нової бази знань;

– procedure button4click(sender: tobject) – додавання нового правила;

– procedure button3click(sender: tobject) – переміщення за правилами назад;

– procedure button2click(sender: tobject) – переміщення за правилами вперед;

– procedure button6click(sender: tobject) – видалення правила;

– procedure spiskishow(sender: tobject) – перегляд списків змінних умови й

виводу;

– procedure tpv1show(sender: tobject) – перегляд таблиці змінних умови;

– procedure tpu1show(sender: tobject) – перегляд таблиці змінних виводу;

– procedure pravshow(sender: tobject) – перехід до перегляду правил;

– procedure pventer(sender: tobject) – редагування змінних виводу;

– procedure zn_pventer(sender: tobject) – редагування значень змінних виводу;

– procedure saveclick(sender: tobject) – збереження бази знань;

– procedure savetofile(name:string) – збереження бази знань у файл;

– procedure exit1click(sender: tobject) – вихід із програми;

– procedure openclick(sender: tobject) – відкриття бази знань;

– procedure sg1getedittext(sender: tobject; acol, arow: integer;

– var value: string) – перевірка на наявність змін у правилі;

– procedure sg1setedittext(sender: tobject; acol, arow: integer;

– const value: string) – перевірка на наявність змін у правилі;

– procedure formclosequery(sender: tobject; var canclose: boolean) – закриття форми;

– procedure n13click(sender: tobject) – вивід інформації про автора;

– procedure n14click(sender: tobject) – вивід довідки;

– procedure sg1keypress(sender: tobject; var key: char) – перевірка на наявність змін у правилі.

Нижче наведені глобальні змінні, використовувані в програмі:

– strok: string; // рядок для відстеження уведених змін;

– path:string; // рядок для імені файлу бази знань;

– t: TextFile; // текстовий файл для завантаження бази знань;

– tek_pr:integer;//номер поточного правила;

– Kol_pr:integer;// кількість правил;

– mp:array of prav;//масив правил;

– tpu: array of uslov; // таблиця змінних умови;

– tpv: array of vyvod; // таблиця змінних виводу;

– pr1,pravilo:prav; // поточне правило;

– dob_pr1,save_pr,izm,otkryt,saved,obn_pr,dob_pr:boolean;

– SPU:array of spu1; SPV:array of spu1; // списки змінних умови й виводу;

– nomer_pu,nomer_pv, nomer_zn_pv:integer; // індекси змінних умови й виводу.

Проект «Klient»

Проект «Klient» служить для роботи експертної системи в режимі консультації. У цьому режимі спілкування з ЕС здійснює кінцевий користувач, якого цікавить результат і (або) спосіб його одержання. У цьому модулі користувач відповідає на питання, запропоновані йому експертною системою. Після одержання відповідей на всі питання, експертна система видає відповідний результат і рекомендацію.

При цьому є можливість подивитися, чому був отриманий даний вивід. У програмі використовуються процедури й функції, які написані для зручності роботи й розуміння програми:

– procedure N2Click(Sender: TObject) – завантажити існуючу базу знань;

– procedure Button1Click(Sender: TObject) – початок роботи системи;

– procedure Button2Click(Sender: TObject) – прийняття відповіді користувача;

– procedure FormCreate(Sender: TObject);

– procedure Button3Click(Sender: TObject) – вивід пояснення до виводу системи;

– procedure N5Click(Sender: TObject) – початку опитування спочатку;

– procedure N3Click(Sender: TObject) – вихід із системи.

Нижче наведені глобальні змінні, використовувані в програмі:

– path:string; // ім'я файлу для відкриття бази;

– t: TextFile; // текстовий файл для відкриття бази;

– mp:array of prav;//масив правил;

– tpu: array of uslov; // таблиця змінних умов;

– tpv: array of vyvod; // таблиця змінних виводу;

– SPU:array of spu1; // список змінних умов;

– SPV:array of spv1; // список змінних виводу;

– nomer_pu,nomer_pv, nomer_zn_pv:integer; //індекси змінних умови й виводу;

– n_a_p, // номер аналізованого правила;

– n_p_u, // номер змінної умови в СПУ;

– n_u, //номер розглянутої умови в аналізованому правилі;

– n_p_v, // номер змінної виводу в СПВ;

– p_u, // індекс першої умови поточного правила;

– kpu, // число умов в аналізованому правилі;

– kpv:integer; // число отриманих виводів;

– z_u:array of string; // масив значень умов із правила;

– z_p:boolean; // значення перевірки поточної умови в правилі z_u_p:array of string; //масив значень умов уведених користувачем;

– Opv:array of string; // черга змінних виводу.

Методичне забезпечення

Дана програма має два модулі «Expert» і «Klient», які використовують у роботі ту саму базу знань.

Інтерфейс модуля «Expert» призначений для експерта в області знань з інформаційної безпеки. За допомогою даного модуля експерт може створювати бази знань, вносити зміни, додавати правила.

Інтерфейс модуля «Klient» призначений для кінцевого користувача. Користувач відповідає на питання, запропоновані йому експертною системою. Після одержання відповідей на всі питання, експертна система видає відповідний результат і рекомендацію, тобто експертна система буде працювати в режимі консультації.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Sinople – симетричний блоковий криптоалгоритм, побудований на основі незбалансованої «мережі Фейстеля». Алгоритм розроблено у 2003 році.

Основні вимоги до алгоритму при його розробці:

– Можливість програмної і апаратної реалізації.

– Висока швидкість.

– Простота.

– Низькі вимоги до пам'яті.

– Високий рівень безпеки.

Алгоритм заснований на 32-розрядних операціях і має 64 раунду, серед яких два типи – C і D.

D раунди спроектовані для досягнення максимальної дифузії, C раунди – для досягнення перемішування. F-функція D раунду використовує один з елементів блоку даних (D[3]) та поточного з'єднання (K[r]) для трансформації 3-х елементів блоку даних.

F-функція C раунду, навпаки, використовує перші три елемента блоку даних і поточний з'єднання (K[r]) для трансформації останнього елемента блоку даних (D[3]).

Раунди D-типу виконуються до раундів C-типу. Додавання ключів з даними проводиться тільки через таблиці замін. Операції XOR (додавання по модулю 2) обов'язково поєднуються з операціями ADD (додавання по модулю 2^{32}).

Таблиці замін спочатку запозичені з алгоритму MARS і містять 512 32-розрядних елементів, проте були жорстко проаналізовано на предмет посилення.

Ключове розклад було спроектовано з урахуванням вимог:

– Простота.

– Використовується та ж процедура, що і при шифруванні та розшифрування.

– Установка ключа займає менше часу, ніж за шифрування.

– Виключення еквівалентних ключів.

– Виключення слабких ключів.

Алгоритм, згідно із заявою авторів, стійкий до лінійного і диференціального аналізу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Дана система призначена для аналізу стійкості криптографічних систем на основі методів штучного інтелекту. Програмне забезпечення розроблялося для персональної обчислювальної техніки не нижче Intel Pentium IV 2500 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.

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».

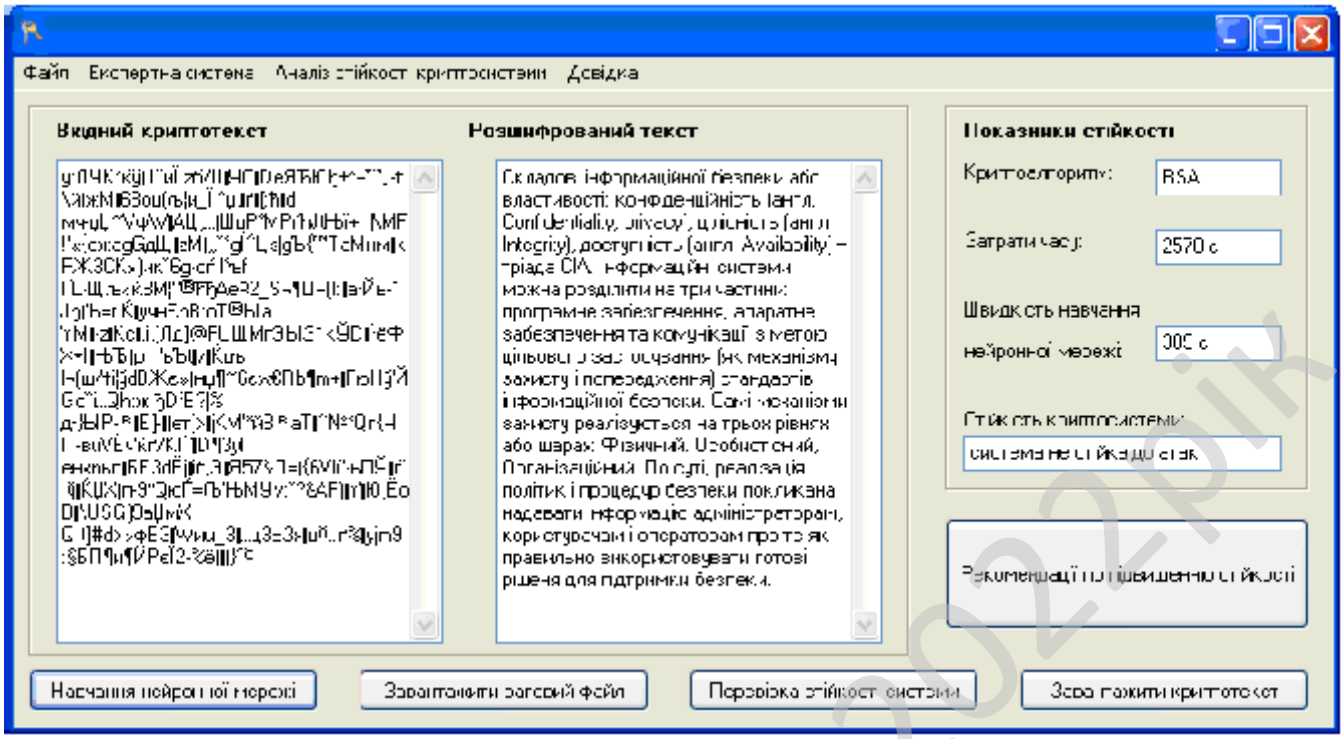


Рисунок 5.1 – Головне вікно програми

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.

На рисунку 5.2 наведено рекомендації по підвищенню стійкості криптосистеми, якщо вона була взламана.

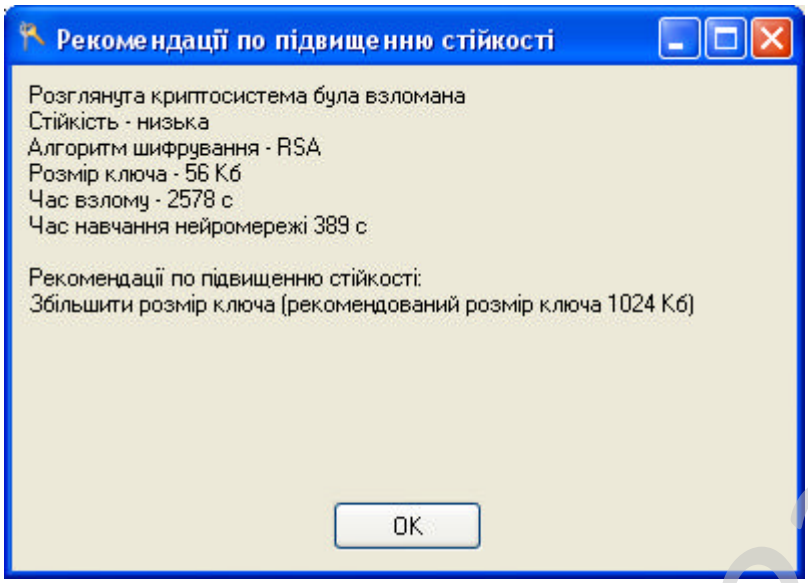


Рисунок 5.2 – Рекомендації по підвищенню стійкості криптосистеми

Переглянути короткі відомості про програму можна натиснувши кнопку «Про програму…» (рисунок 5.3).

Рисунок 5.3 – Вікно довідки

6 НАУКОВА НОВИЗНА

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

Метою розробки є дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів.

Об'єктом дослідження є процес формування профілів захисту хмарних сервісів.

Предметом дослідження є методи формування профілів захисту хмарних сервісів.

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

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

– Удосконалено метод формування профілів захисту хмарних сервісів.

– Розроблено вітчизняний продукт формування профілів захисту хмарних сервісів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі була досліджена та розроблена програмна реалізація системи формування профілів захисту хмарних сервісів.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	20
3. Запланований термін розробки, днів	Fpq	48 (2 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)	–	2
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.	–	20000
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: W_i – 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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 68 = 114 \text{ люд/день.}$$

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	114	Ф 7.1-7.4
Впровадження	13	Д13
Всього	155	–

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.

$$Ч = \frac{155 \cdot 1}{48 - 4} = 3,5 \text{ ставки.}$$

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	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	150	375	6,25
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	49,91

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{50 \cdot 2}{1{,}2} = 83 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 83/(48 \cdot 8) = 0{,}2 \text{ ставка.}$$

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

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

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

Посада	Вид роботи	Час	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	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,5	26693	26693
Продакт-менеджер	0,25	20000	10000
Інженер-програміст	3,5	25000	175000
Інженер-електронщик	0,2	20000	8000
Інженер-системотехнік	0,25	20000	10000
Адміністратор мережі	0,5	20000	20000
Системний програміст	0,25	20000	10000
Дизайнер WEB	0,25	20000	10000
Інженер-верстальник	0,25	20000	10000
Бухгалтер-економіст	0,5	20000	20000
Всього за період розробки	$R_{сп} = 6,45$	-	$Ф_{роб} = 299693$

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.

$$З_{сд} = \frac{299693}{6,45 \cdot 48} = 968 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

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

$S_{у}$ – 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:

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

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 Інтернет-магазину Компбест за 16.10.22 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок Нettоп Fujitsu Esprimo Q556 USFF		7347
Процесор	Intel Core i3-6100T (2 (4) ядра по 3.2 GHz), MB Smart Cache	-
Системна плата	Материнська плата Intel H110 Chipset, 4 USB 2.0, 4x USB 3.0, 1x DVI, 2x DisplayPor 1x COM-порт, 3x Audio, 1x Ethernet	-
Відеокарта	Інтегрована Intel HD Graphics 530 (до 179 MB)	-
Жорсткий диск	120 GB SSD + Seagate Barracuda 7200.1 500GB 7200rpm 16MB ST500DM002 3. SATA III	-
Оперативна пам’ять	Kingston 4 GB DDR4	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	Fujitsu Esprimo Q556 USFF, PSU 350W(FS Brand: ATX-350PNR, 12 cm), black, (fron bezel – black+light silver; body material 0.6mm), 80mm fan (rear) 2xUSB3.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-
Кулер	–	–

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
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) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

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

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
Нематеріальні активи			
4. Нематеріальні активи	20000	10	2000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	143000	20	28600
7. Господарський інвентар	28000	25	7000
Всього по групі	180031	-	5000
Разом	$К_р$ = 1807208		$А_р$ = 176988,5

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн.

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.

$$З_о = 968 \cdot 155/20 = 7503 \text{ грн.}$$

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, %.

$$З_д = 7503 \cdot 10 \cdot 0{,}01 = 750{,}3 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(7503+750{,}3) = 3054 \text{ грн.}$$

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, %.

$$Г_{осп} = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

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 3):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 37,33 грн./шт.

$$З_{М2} = 37{,}33 \cdot 3 = 112 \text{ грн.}$$

Згідно виданих викладачем 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.

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

$$З_М = (105 + 112 + 1702)/20 = 96 \text{ грн.}$$

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, %.

$$О_п = 7503 \cdot 15 \cdot 0{,}01 = 1125 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 20 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.

$$A_м = 176989 \cdot 2/(20 \cdot 12) = 1475 \text{ грн.}$$

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

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

$$C_п = 7305+730,5+3054+1125+96+1125+1475 = 14910,5 \text{ грн.}$$

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	2	3
1. Основна зарплата виконавців	$З_о$	7305
2. Додаткова зарплата виконавців	$З_д$	730,5
3. Відрахування на соціальні потреби	$C_{оц}$	3054
4. Загальногосподарські витрати	$Г_{осп}$	1125
5. Витрати на матеріали	$З_М$	96
6. Освоєння нових операційних систем, мов програмування	$О_п$	1125
7. Амортизація основних фондів	$A_м$	1475
8. Повна собівартість програмного забезпечення	$C_п$	14910,5
9. Плановий прибуток	$П_р$	7455,25
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	22365,75
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4473,15
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	26838,9

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 14910{,}5 = 7455{,}25 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	26839
Всього капітальних витрат	–	26839

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	$З_р$	34892	8723
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	6710
Всього витрат за рік	$І$	34892	15433

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування сервера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилось з 400 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 400 \cdot 65 \cdot 1{,}1 \cdot 1{,}22 = 34892 \text{ грн},$$

до:

$$З_{р\,нов} = 100 \cdot 65 \cdot 1{,}1 \cdot 1{,}22 = 8723 \text{ грн}.$$

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

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

Визначити різницю споживання електроенергії при впровадженні систем не має можливості, тому витрати на електроенергію в розрахунку приймаємо рівними нулю.

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	–	26839	–	6709,75
Всього відрахувань	-	–	26839	–	6709,75

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.

$$Ев = (22365 - 14910) \cdot 20 - (0,05 \cdot 1408000 + 0,5 \cdot 199177 + +0,2 \cdot 143000 + 0,25 \cdot 37031 + 0,1 \cdot 20000) \cdot 2/12 = 104286 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{399208}{(22365 - 14910) \cdot 20 \cdot 12 / 2} = 0{,}5 \text{ років}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1	2	3
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	14910
3. Ціна розробленої програми	Грн.	22365
4. Плановий прибуток від реалізації розробленої програми	Грн.	7455
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1807208
7. Загальний прибуток від реалізації програмної продукції	Грн.	149100
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	104286
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	26839
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12749
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	1,4

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

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.

$$E_{сп} = (34892\text{-}15433)\text{-}0{,}25 \cdot 26839 = 12749 \text{ грн.}$$

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

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

$$T_{сп} = \frac{26839}{34892 - 15433} = 1{,}4 \ роки.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

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

Комп'ютерні мережі вриваються у життя людей в їх професійну діяльність найнесподіванішим і масовим чином. Вони породили істотно нові технології обробки інформації – мережні технології. У найпростішому разі мережні технології дозволяють спільно використовувати ресурси – нагромаджувачі великий ємності, друкують устрою, доступ в Internet, бази й банки даних. Найбільш сучасні і перспективні підходи до мереж пов'язані з допомогою колективного поділу праці – розробці різних документів і майже проектів, управлінні установою чи підприємством, і т.д.

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Працівники, використовують комп'ютерну техніку, на своєму досвіді оцінили її величезні можливості. Одночасно виникла певна безтурботність при її експлуатації.

Недотримання вимог безпеки призводить до того, що й через кілька днів роботи за комп'ютером співробітник починає відчувати певний дискомфорт: в нього виникає головний біль і різь у власних очах, з'являються почуття виснаження й дратівливість. В окремих людей порушується сон, погіршується зір, занедужують руки, шия, поперек тощо.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст.14 Закону «Про охорони праці» [1] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці.

8.2 Аналіз умов праці

Приміщення розташовано на третьому поверсі п'ятиповерхового будинку. У приміщенні розташовано 3 робочих місць з комп'ютерами (далі ПК). Відповідно до норм «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2] площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м^2, об'єм не менше 20 м^3. Розміри даного приміщень складають: довжина – 6 м, ширина – 4,5 м, висота – 3,5 м, тобто загальна фактична площа складає 27 м^2. Необхідна площа на 3 робочих місця із установленими ПК складає 18 м^2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 31,5м^3, отже відповідає нормі ДСанПіН 3.3.2-007-98 – не менше 20 м^3.

Таблиця 8.1 – Перелік шкідливих та небезпечних виробничих факторів

Найменування факторів	Можливі джерела їх виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що згорають і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Іонізація повітря	Статична електрика випромінювання	Шкідливий
Підвищений рівень шуму	Шум створюється перетворювачем напруги ЕОМ, її технічною периферією, а також людьми, що працюють в приміщенні	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Незадовільні параметри мікроклімату	Незадовільний стан системи опалення і вентиляції	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

При роботі з ПК людина може піддатися впливу шкідливих та небезпечних факторів. Під шкідливими виробничими факторами розуміють фактори, тривалий вплив яких викликає розвиток професійних захворювань. Небезпечні виробничі

фактори – вплив яких на працюючого викликає травму, тобто пошкодження організму. Шкідливі і небезпечні чинники, з якими стикається бібліограф при роботі з ПК, приведені в таблиці 8.1.

По категорії вибухо- і пожежонебезпеки, згідно дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір і інше. Виходячи з категорії пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» [3] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

По ступені небезпеки поразки людей електричним струмом відділ, згідно, класифікується як приміщення з підвищеною небезпекою, тому що не виключена можливість одночасного дотику людини до маючих з'єднання з землею конструкціям будинку, з одного боку, і до металевих корпусів електроустаткування, що можуть виявити під напругою – з іншого.

Для забезпечення вищевказаних оптимальних метеорологічних умов у помешканні передбачена система опалення (загальне парове) в холодному періоді, та вентиляція і кондиціювання в теплий період року, згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціювання» [4]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно – обчислювальних машин».

Припустимий рівень іонізації повітря помешкання відповідно до СН 21.52-80 повинен складати 1500 – 3000 один./м^3.

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [5]

Відділ забезпечений комбінованим освітленням. В темний час доби передбачається загальне і/або місцеве рівномірне штучне, а в світлий – бокове одностороннє природне освітлення два віконних прорізи.

Одним з найбільш поширеніших чинників зовнішнього середовища, який несприятливо впливає на людину, є шум. Вплив шуму на організм людини залежить від рівня звукового тиску, частотних характеристик, тривалості дії, а також індивідуальних особливостей людини.

При тривалій дії шуму у оператора ЕОМ виявляються симптоми утомленості, нервового збудження, що сприяють погіршенню працездатності і допущенні помилок при роботі. Для уникнення шкідливої дії шуму на організм працюючого, необхідне дотримання нормованих параметрів, які не повинні перевищувати допустимих величин. При роботі на комп'ютері рівень шуму не повинен перевищувати 50 дБА. Приміщення розташоване вікнами у двір і знаходиться далеко від проїжджої частини вулиці. Основними джерелами шуму в приміщенні є устаткування і люди. Розглянута кімната не призначена для прийому відвідувачів і тому в ній не спостерігається великого скупчення людей. Тому основним джерелом шуму є комп'ютерна техніка.

Джерелами шуму при роботі ЕОМ є механічні частини принтера, що рухаються, і вентилятори(Lпк = 35 дБА, , Lprn= 48 дБА.) При роботі вентиляційної системи, що забезпечує оптимальний температурний режим електронних блоків ЕОМ і вмонтована в задню панель, створюється аеродинамічний шум. Шум, створюваний працюючим комп'ютером, може бути охарактеризований як широко смуговий постійний з аперіодичним посиленням при роботі принтера. Час роботи ПЕОМ – 6 – 8 год. за добу; принтери працюють не більш 1,5-2 год. за добу.

При наявності великої кількості джерел шуму еквівалентне значення шуму $L_{ЭКВ}$, дБА розраховують по наступній формулі:

$$L_{ЭКВ} = 10 \cdot \lg\left(\frac{1}{T}\sum_{i=1}^{n}\left(t_i \cdot 10^{0.1 \cdot L_i}\right)\right) \quad (8.1)$$

де

L_i – рівень шуму і-го джерела (пристрою);

t_i – час роботи і-го джерела (пристрою);

T – загальний час роботи;

n – кількість джерел шуму даного типу;

Для даного приміщення необхідні змінні складають:

Загальний час роботи – робітник день, тобто T=8 годин.

Для фонового шуму (вентиляторів):

L_1 = 35 дБА, T_1 = 8 годин, n_1=15 (5x3);

Для лазерного принтера Lexmark Jet:

L_2 = 48дБА, T2 = 2 години, n_2=1, для сканера L_3 = 46дБА, T3 = 2 години.

Підставляємо отримані величини у формулу (8.1):

$$L_{экв} = 10 \cdot \lg\left(\frac{1}{8} \cdot \left(15 \cdot 8 \cdot 10^{0,1 \cdot 35} + 1 \cdot 2 \cdot 10^{0,1 \cdot 48} + 1 \cdot 2 \cdot 10^{0,1 \cdot 46}\right)\right) = 46,3 \text{ дБА}$$

Таким чином, еквівалентний рівень шуму в приміщенні за робітник день Lэкв = 46,3 дБА , тобто не перевищує норму 50 дБА.

8.3 Техніка безпеки та протипожежна профілактика

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 V. Про це є відповідні написи на кожному розподільному щиті.

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі

роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне: достатнє висвітлення екрана дисплея й робочого місця; повна технічна справність устаткування, його електробезпечність; достатня пожежобезпечність приміщення; оптимальний мікроклімат, що сприяє продуктивній роботі; відповідність робочого місця вимогам ергономіки. До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки електричним струмом, при електроні справності встаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими параметрами; робота при недостатній освітленості екрана дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [6], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

8.4 Розробка заходів з охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

Для зменшення шуму в приміщенні пропоную використовувати замість матричного принтера, що створює багато шуму, більш тихий – лазерний принтер.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості, проведених відділом охорони праці.

Як міри по зниженню шуму можна запропонувати:

- облицювання стелі і стін звукопоглинаючим матеріалом (знижують шум на 6-8 до);

- екранування робочого місця (постановкою перегородок, діафрагм);
- установка в комп'ютерних приміщеннях устаткування, що робить мінімальний шум;
- раціональне планування приміщення.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення вузла, у випадку пожежі, повішену на вхідні двері.

8.5 Висновки до розділу

У даному розділі магістерської роботи були виконано аналіз умов праці користувачів ПК, які працюють у зазначеному приміщенні Проведено перевірка організації робочого місця із відповідними замірами параметрів мікроклімату, освітлення, рівня шуму та розрахунком рівня шуму.

Розроблені заходи щодо поліпшення умов праці дотримання техніки безпеки та проведення протипожежної профілактики дозволить створити умови, які будуть забезпечувати більш комфортну роботу.

9 ОСНОВНІ ВИСНОВКИ

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

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

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

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем формування профілів захисту хмарних сервісів.

– Досліджена система формування профілів захисту хмарних сервісів.

– На основі отриманих результатів досліджень створена програмна реалізація системи формування профілів захисту хмарних сервісів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання формування профілів захисту хмарних сервісів.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Sinople.

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 12749 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,4 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ященко Д.Р. Дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». CEUR Workshop Proceedings Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe (Scopus).

3. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». CEUR Workshop Proceedings Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe (Scopus).

4. Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

5. Kovalenko Oleksandr, The mathematical model of the testing technology for DOM XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

6. Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific &

practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

7. Коваленко О.В. Моделі та методи розроблення програмного забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

8. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

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

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

11. Коваленко А.В. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

12. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

13. Коваленко А.В. Проблемы анализа и оценки рисков информационной

деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

14. Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск 2(23). – Харків: ХУПС. – 2016. – С. 150-158.

15. Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

16. Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

17. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

18. Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

19. Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

20. Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

21. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління,

навігації та зв’язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

22. Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький: ЦНТУ. – 2018. – С. 128-140.

23. Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

24. Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

25. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

26. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

27. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

28. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

29. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

30. Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

31. Коваленко О.В. Математична модель технології тестування вразливості до SQL ін’єкцій / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

32. Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

33. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

34. Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

35. Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.В. Коваленко, А.А. Смирнов // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

36. Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”.

м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

37. Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

38. Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

39. Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

40. Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез VIII міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

41. Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

42. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А.

Смирнов, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

43. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

44. Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.В. Коваленко, А.А. Смирнов // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

45. Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

46. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

47. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea

informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

48. Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез дев'ятнадцятого міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

49. Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

50. Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. Кіровоград: КНТУ. – 2017. – С. 92.

51. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

52. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces "Alexandru cel Bun". – 2017. – P. 7.

53. Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп'ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

54. Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

55. Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

56. Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

57. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

58. Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов

// Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

59. Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

60. Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп’ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада

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

62. 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

63. 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

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

65. 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

66. 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

67. 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

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

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

70. 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

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи формування профілів захисту хмарних сервісів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи формування профілів захисту хмарних сервісів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці.

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 – 107 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 23.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація системи формування профілів захисту хмарних сервісів

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 Секунд,  ', S) +
              FormatFloat('#0 ms,   ', S * 1000) +
              Format('%d байт, %s Kb розмір даних  ', [Size,
FormatFloat('#,##0.00', Розмір / 1024)]) +
              FormatFloat('са #,##0.00 Мб/с', (1 / S) * (Розмір / (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, 'Самотестування не пройдене', 'Самотестування хеш-
функції', 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.Варіант }
//  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, 'Самотестування не пройдене', 'Самотестування алгоритму
шифрування', 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 =
    ('Шифрування тексту', 'Шифрування ланцюжка блоків ', 'Шифрування зі
зворотнім зв'язком ',
     'Зворотній зв'язк по виходу ', 'Електроний кодовий блокнот', '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('Пароль', 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) + ' Сектор заповнений
зашифрованими даними.',
              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 := 'Тестовий вектор для ' + 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 до буферу обміну}
```

```
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 := 'Тестовий вектор для ' + 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; // використовує формат рядка
    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('Тест', nil, Format);
  DoInfo('MD5'#9+S, clWindowText);

  S := HUser.CalcString('Тест', 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 := 'Пароль';
    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 Частина 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 := 'Пароль';
    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 Байт з буфера
    Calc(Buffer[33], 15);   // розраховуємо з  Buffer[33] 15 байт
    for I := 1 to 11 do
      Calc('DEC Частина 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», Пароль «DEC"',
clGreen);
    Protection := TMAC_RFC2104.Create('DEC', nil); // Ваш пароль

    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», Пароль «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); // Формат повинен
бути fmtCOPY
    DoInfo('2. encoded'#9+StrToFormat(PChar(S), Length(S), Format),
clWindowText);

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

//---------------------------------------------------------------------------
// paScramble, це одношляхова функція
    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);

//---------------------------------------------------------------------------
// розраховуємо MD5 Fingerprint над Blowfish зашифрований DEMO.EXE
// THash_MD5.CalcFile() с  Blowfish шифром розраховується
// MD5 автентифікатор.
// Цей метод Взламуванняує DEMO.EXE, розраховуючи MD5 та Взламуванняує MD5 Final
Digest
    DoInfo('MD5 розраховується над Blowfish Взламуванняючи 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 Взламуванняаним рядком
    DoInfo('MD5 розрахований над Blowfish Взламуванняаним рядком ', clGreen);
// використовуємо TProtection методи для побудови ланцюжка
    Protection := TCipher_Blowfish.Create('DEC Частина 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 Взламуванняуємо рядок
      Cipher.InitKey('DEC Частина 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
      // Взламуванняуємо 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 Взламуванняу
  FileName := ParamStr(0); // вибираємо файл для Взламуванняу

  DoInfo('Хеш повідомлення автентифікаційного 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 Частина I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC Частина I"'#9+S, clWindowText);

  S := THash_MD5.CalcFile(FileName, TMAC.Create('DEC Частина I', nil), Format);
  DoInfo('MAC-MD5, Пароль "DEC Частина 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 з захистом,
THash_XXX(TMAC("DEC", TCipher_Blowfish(«Зашифрований текст»)))', clBlue);
// визначіть  Blowfish Protection, це кодує фінальний Hash.DigestKey
    MAC.Protection := TCipher_Blowfish.Create('Зашифрований текст', 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 захищений
//  TRandom_LFSR з періодом 2^400-1, дивіться RNG.pas якщо потрібна додаткова
інформація
    MAC.Protection := TRandom_LFSR.Create('Зашифрований текст', 400, False,
nil);

    DoInfo('4. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", TRandom_LFSR(«Зашифрований текст»)))', 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) захист
// a Double HMAC -> HMAC-MD5-HMAC-MD4
    MAC.Protection := THash_MD4.Create(TMAC.Create('Зашифрований текст', nil));
// Ланцюжок: THash_XXX -> TMAC('DEC') -> THash_MD4 -> TMAC('Зашифрований текст')
    DoInfo('5. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_MD4(TMAC(«Зашифрований текст»))))', 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('Зашифрований текст', nil);

    DoInfo('6. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_MD4(TMAC(«Зашифрований текст»))))', 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(«Зашифрований текст»))
// a HMAC-MD5-HMAC-SHA1
```

```
    MAC.Protection := THash_SHA1.Create(TMAC.Create('Зашифрований текст', nil));

    DoInfo('7. Загальний TMAC -> використовує TMAC Instance з захистом,
THash_XXX(TMAC("DEC", THash_SHA1(TMAC(«Зашифрований текст»)))', 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(«Зашифрований
текст»))', clBlue);

  Protection := TCipher_Blowfish.Create('Зашифрований текст', 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(«Зашифрований
текст»))', clBlue);

  Protection := TRandom_LFSR.Create('Зашифрований текст', 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('Тут ', TMAC_RFC2104.Create(RepKey($0B, 16), nil),
fmtHEXL);
  DoInfo('HMAC-MD5'#9+S, clWindowText);
  DoInfo(#9'9294727a3638bb1c13f48ef8158bfc9d', clGrayText);

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

  S := HUser.CalcString('Тут ', 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('нічого не робити?', MAC, fmtHEXL);
    DoInfo('HMAC-MD5'#9+S, clWindowText);
    DoInfo(#9'750c783e6ab0b503eaa86e310a5db738', clGrayText);

    S := THash_SHA1.CalcString('нічого не робити?', MAC, fmtHEXL);
    DoInfo('HMAC-SHA1'#9+S, clWindowText);
    DoInfo(#9'effcdf6ae5eb2fa2d27416d5f184df9c259a7c79', clGrayText);

    S := HUser.CalcString('нічого не робити?', 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 Біт розділених по 8 біт * 2 Chars для байта
```

```
  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 добрий ', clGreen);
// зберігаємо поточний TAN в Database та останній клієнтський TAN
      LastTAN := TAN;
      DoInfo('останній TAN:'#9+LastTAN, clGreen);
    end else
    begin
      DoInfo('TAN поганий', clMaroon);
    end;
    DoInfo('', clWindowText);

// 2. TA ---------------------------------------------------------------------
    TAN := TANList[0]; TANList.Delete(0);
    DoInfo('поточний TAN:'#9 + TAN, clWindowText);
    Delete(TAN, 1, 4);
    DoInfo('Поганий 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 добрий ', 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 Взламуванняу

  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 шифрування рядка"', clBlue);
    InitKey('DEC Частина I', nil);

    S := EncodeString('Тест дешифрування 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 Частина 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('Взламування 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 Частина 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('Взламування 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('Взламування 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('Взламування 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('Взламування 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('Взламування 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('Взламування 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 Частина 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 Взламуванняу

  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() у MAC режимі: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() у MAC режимі: ' + 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(«Зашифрований
текст»)', clBlue);
  with CUser.Create('DEC', TCipher_Blowfish.Create('Зашифрований текст', nil))
do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() у MAC режимі: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() у MAC режимі: ' + 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('Захищено HMAC-MD5-LFSR128-SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                      TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для MD5', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                       TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для LFSR', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
```

```
                        TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено HMAC-MD5-LFSR128-SCOP з паролем для SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Зашифровані дані 1',
                    TRandom_LFSR.Create('Зашифровані дані 2', 128, False,
                       TCipher_SCOP.Create('Зашифровані дані 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Частина 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 з періодом 2^400-1', clBlue);

  with TRandom_LFSR.Create('', 400, False, nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC Частина I"', clBlue);
    Seed('DEC Частина 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 Частина I"', clBlue);
    Seed('DEC Частина 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 зі збереженням у State', clBlue);
    DoInfo('Ключ на період  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 Частина 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 Частина 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 Частина 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 Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

    S := CodeString('Добрий DEC Частина I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

end.
```

Chiper.pas – файл реалізації алгоритмів для криптоаналізу 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      Шифрування тексту
  cmCBC      Шифрування ланцюжка блоків
  cmCFB      K-bit Шифрування зі зворотнім зв'язком
  cmOFB      K-bit Зворотній зв'язк по виходу
  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;
//!!
  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  // спеціально для 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       // Це 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         // це 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 // спеціально для 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     // це прискорюється для AMD Chips,
//        MOV     CL,BH    // це прискорюється для 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  // Користувач
         MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди

@@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  // Користувач
          MOV   EDI,TEA_Delta
          MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди
          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.
```