МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Кузьменко Єгор Олексійович

Програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access

Спеціальність: 125 «Кібербезпека»

Освітній ступінь: бакалавр

Науковий керівник:

Буравченко Костянтин Олегович _____

(підпис) (дата)

кандидат технічних наук

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Кузьменку Єгору Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access*

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

затверджені наказом вищого навчального закладу № 185-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

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

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кузьменко Є.О. Програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи кібербезпеки DMZ з використанням технології Reverse Access.

Метою розробки є програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access.

Результат роботи – програмна реалізація системи кібербезпеки DMZ з використанням технології Reverse Access.

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 XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: кібербезпека, DMZ, Reverse Access

ABSTRACT

Kuzmenko Ye.O. DMZ cybersecurity system software using Reverse Access technology. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This bachelor's degree software has been developed for the DMZ cybersecurity system using Reverse Access technology.

The purpose of the development is the software of the DMZ cybersecurity system using Reverse Access technology.

The result is a software implementation of the DMZ cybersecurity system using Reverse Access technology.

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.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment of Delphi 10.4 Sydney.

Keywords: cybersecurity, DMZ, Reverse Access

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DMZ	–	демілітаризована зона – Demilitarized Zone – це спеціальний сегмент локальної мережі в який виводяться сервіси, до яких повинен бути відкритий повний доступ як із внутрішньої мережі, так і із зовнішньої
ICMP	–	міжмережевий протокол управляючих повідомлень
IP	–	Internet Protocol – міжмережевий протокол
LAN	–	локальна мережа організації
RFC	–	опис набору протоколів Internet
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
TCP	–	Transmission Control Protocol – протокол управління передачею
UDP	–	User Datagram Protocol – протокол користувальницьких датаграм
URL	–	уніфікований покажчик інформаційного ресурсу

ВСТУП

Актуальність теми. Демілітаризована зона – DMZ (Demilitarized Zone) – це спеціальний сегмент локальної мережі в який виводяться сервіси, до яких повинен бути відкритий повний доступ як із внутрішньої мережі, так і із зовнішньої. При цьому приватна мережа як і раніше закрита за роутером. Ніяких відмінностей у її роботі не буде. А от DMZ-хост тепер повністю доступний з Інтернету й забезпечує свою безпеку сам. Іншими словами всі його відкриті порти видні з поза по «білій» IP-адресі. У випадку з реєстратором досить просто змінити пароль, що використовується по-умовчанню, а от якщо це ігровий сервер, то варто приділити особливу увагу налаштуванням міжмережевого екрана.

Прості недорогі маршрутизатори організувати повноцінну Демілітаризовану Зону для цілого сегмента не можуть, так від пристроїв цього класу такого й не потрібно. Зате вони дозволяють вивести в неї тільки один з вузлів мережі, зробивши з нього DMZ-хост, відкривши в зовнішню мережу всі його доступні порти. А нам це й потрібно! Як це зробити? Запускаємо браузер, уводимо IP-адресу роутера (звичайно це 192.168.1.1 або 192.168.0.1) і попадаємо у веб-конфігуратор. А далі треба в меню знайти розділ « DMZ». У пристроїв від Asus – це вкладка в розділі «Інтернет»:

У роутеров D-Link ця функція перебуває в Міжмережевому екрані. На деяких моделях, в Zyxel Keenetic наприклад, вона може бути розташована в параметрах NAT. Алгоритм подальших дій простий – треба включити функцію, поставивши відповідну галочку. Нижче буде поле, у яке необхідно ввести IP-адресу сервера, комп'ютера або відео-реєстратора, який ми виведемо в DMZ. Застосовуємо налаштування. Готове.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access.

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

– Огляд існуючих систем кібербезпеки DMZ з використанням технології Reverse Access.

– Дослідження системи кібербезпеки DMZ з використанням технології Reverse Access.

– Програмна реалізація системи кібербезпеки DMZ з використанням технології Reverse Access.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі DMZ з використанням технології Reverse Access.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

В області комп'ютерної безпеки DMZ (іноді називана мережею периметра) – це фізична або логічна підмережа, яка містить зовнішні служби організації й надає їхній більшій мережі, що не довіряється, звичайно Інтернету. Ціль DMZ – додати додатковий рівень безпеки в локальну мережу організації (LAN); зовнішній зловмисник має доступ тільки до устаткування в демілітаризованій зоні, а не до якої-небудь іншої частини мережі. Назва походить від терміна «демілітаризована зона», територія між національними державами, у якій воєнні дії заборонені.

Звичайною практикою є наявність брандмауера й демілітаризованої зони (DMZ) у вашій мережі, але багато з людей, навіть IT-фахівці, не зовсім розуміють, чому, за винятком деякої розпливчастої вистави про стат-безпеку.

Більшість підприємств, які розміщають свої власні сервери, управляють своїми мережами з DMZ, розташованої по периметру їх мережі, що звичайно працює на окремому брандмауерові в якості зони з напівжорсткою довірою для систем, які взаємодіють із зовнішнім світом.

Щоб підтримувати реальну безпеку, важливо чітко розуміти призначення DMZ.

Більшість брандмауерів є пристроями безпеки мережевого рівня, звичайно цей пристрій або пристрій у комбінації з мережевим устаткуванням. Вони призначені для надання деталізованих засобів контролю доступу в ключовій точці ділової мережі. DMZ – це область вашої мережі, яка відділена від вашої внутрішньої мережі й Інтернету, але підключена до обох.

DMZ призначена для розміщення систем, які повинні бути доступні в Інтернеті, але не так, як ваша внутрішня мережа. Ступінь доступності Інтернету

на рівні мережі контролюється брандмауером. Ступінь доступності Інтернету на рівні застосунків контролюється програмним забезпеченням і, по суті, комбінацією веб-сервера, операційної системи, користувацького застосунку й часто програмного забезпечення бази даних.

1.2 Область застосування

DMZ звичайно дозволяє обмежений доступ з Інтернету й із внутрішньої мережі. Внутрішні користувачі звичайно повинні одержувати доступ до систем у демілітаризованій зоні, щоб обновляти інформацію або використовувати дані, зібрані або оброблені там. DMZ призначена для надання громадськості доступу до інформації через Інтернет, але обмеженим образом. Але оскільки Інтернет і мир геніальних людей піддаються ризику, існує постійний ризик того, що ці системи можуть бути скомпрометовані.

Вплив компромісу має два аспекти: по-перше, інформація про вразливі системи може бути загублена (тобто скопійована, знищена або ушкоджена), а по-друге, сама система може використовуватися в якості платформи для подальших атак на чутливі внутрішні системи.

Щоб зменшити перший ризик, DMZ повинен дозволяти доступ тільки через обмежені протоколи (наприклад, HTTP для звичайного веб-доступу й HTTPS для зашифрованого веб-доступу). Потім самі системи повинні бути ретельно налаштовані для забезпечення захисту за допомогою дозволів, механізмів автентифікації, ретельного програмування, а іноді й шифрування.

Подумайте, яку інформацію буде збирати й зберігати ваш веб-сайт або застосунок. Це те, що може бути загублене, якщо системи скомпрометовані за допомогою звичайних веб-атак, таких як SQL-ін'єкція, переповнення буфера або неправильні дозволи.

Щоб зменшити другий ризик, системи DMZ не повинні довірятися системам, розташованим глибше у внутрішній мережі. Інакше кажучи, системи

DMZ не повинні нічого знати про внутрішні системи, хоча деякі внутрішні системи можуть знати про системи DMZ. Крім того, засоби керування доступом до DMZ не повинні дозволяти системам DMZ установлювати які-небудь додаткові з'єднання в мережі. Замість цього будь-який контакт із системами DMZ повинен ініціюватися внутрішніми системами. Якщо система DMZ скомпрометована як платформа для атаки, єдиними видимими їй системами повинні бути інші системи DMZ.

Надто важливо, щоб IT-менеджери й власники бізнесу розуміли типи можливих ушкоджень систем, підданих впливу Інтернету, а також механізми й методи захисту, такі як DMZ. Власники й менеджери можуть ухвалювати обґрунтовані рішення про те, які ризики вони готові прийняти, тільки коли вони твердо розуміють, наскільки ефективно їх інструменти й процеси знижують ці ризики.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Видалося б, вибір міжмережевого екрану, або як його ще називають, файєрволу для Windows – справа не складна: установлюй так користуйся. Але файєрволи для ПК бувають різні, і вибір може коштувати зусиль і часу. У цій статті ми розглянемо 8 файєрволів і перевіримо їхній можливості й первісні налаштування за допомогою простого функціонального тесту.

Найбільш важливі фактори при виборі файєрволу для Windows – це простота й легкість налаштування, наявність навчального режиму, коли файєрвол діє на нерви задає купу питань при спробах вийти в мережу для кожного ПЗ, а все інше забороняє. Ще, мабуть, важливі наявність російського інтерфейсу й безкоштовність, а також додаткові функції.

Порівнювати будемо наступні програми:

- Comodo Firewall.
- Avast! Internet Security.
- AVG Internet Security.
- Outpost Firewall Pro.
- Zonealarm Free Firewall.
- Privatefirewall.
- Glasswire.
- Tinywall.

Усі ці програми позиціонується розроблювачами як засобу захисту для домашніх комп'ютерів, тому ми не будемо залазити в нетрі налаштувань і порівнювати ці програми по функціях і різним модулям. Краще подивитися на них очима простого користувача.

Звичайний користувач відрізняється від просунутого чисто утилітарним підходом до програмного забезпечення: нажали «Установити» і віримо в чудо автоматизації, яке врятує й захистить від хакерів з скриптами й троянами. А як там воно влаштоване усередині, більшості зовсім не важливо.

Тому умови експерименту застосунали максимально спростити. Ми встановимо кожний з файєрволів на чисту ОС Windows 10 x64 і спробуємо запустити одну програму, яка починає ломитися на зовнішній сервер, імітуючи підозрілу мережеву активність. Потім ми включимо режим навчання й повторимо тест знову. Нарешті, на третьому етапі ми настроїмо файєрвол на основі білого списку, заборонивши йому все, що явно не дозволене.

Ну й не будемо забувати про важливі для нас критерії: вартість ліцензії, мова інтерфейсу й простоту налаштування й установки. А отримані результати ми потім зрівняємо.

Comodo Firewall

- Офіційний сайт: comodo.com.
- Системні вимоги: Windows XP SP2, Vista, 7, 8, 10, 152 Мбайт RAM, 400 Мбайт на диску.
- Ліцензія: є безкоштовна версія й Pro.
- Вартість: безкоштовно або 39,99 долара за Pro.
- Мова інтерфейсу: кирилиця.

Ця програма одержала широку популярність ще в епоху Windows XP, коли Comodo Firewall був чи не найпоширенішим безкоштовним файєрволом в Україні. Користується він популярністю й зараз. Що, у загальному-то, не дивно: розроблювачі обіцяють нам проактивний захист із HIPS, міжмережеве екранування, захист від переповнення буфера й несанкціонованого доступу, захист реєстру й системних файлів, а також інші смачні булочки.

Однак під час установки файєрвол викликав змішані почуття. Спочатку пропонував поставити розширення для Яндекс.Браузера.

А потім, якщо не звернути увагу на «компоненти» і не виключити все непотрібне, установник інсталює на ваш комп'ютер свій браузер.

Робимо тест, і Comodo пропускає утиліту для тестування файєрволів Firewall Tester.

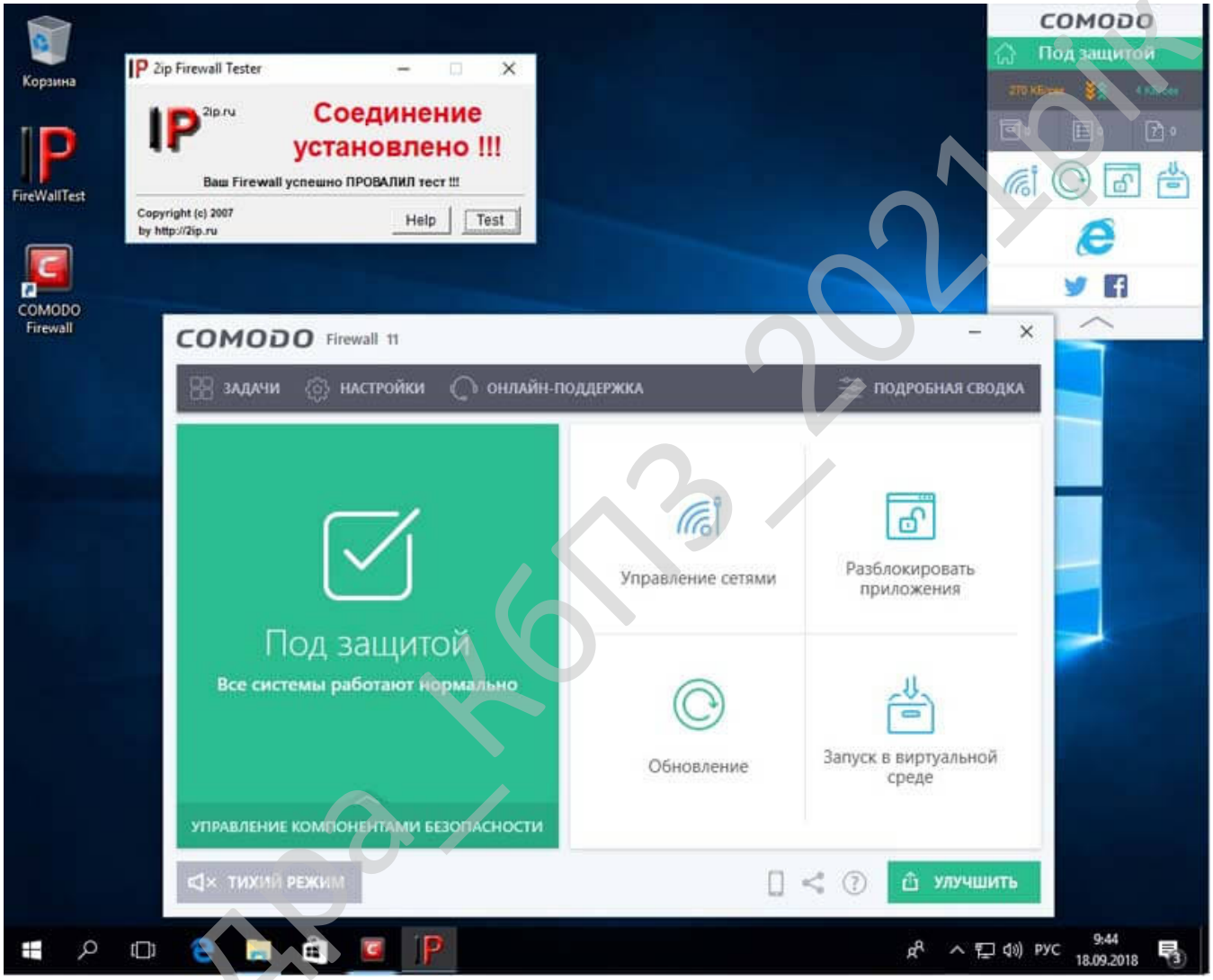

Рисунок 2.1 – Тест Comodo Firewall

Включили режим навчання в налаштуваннях – і файєрвол чомусь ніяк не прореагував на нашу тестову програму, яка успішно підключилася до віддаленого комп'ютера.

Тільки після складання білого списку нарешті вдалося заблокувати Firewall Tester. Вивід напрошується суперечливий: Comodo Firewall – дуже відомий файєрвол, але установка непотрібного програмного забезпечення псує все враження. А результати тесту виявилися сумними: для забезпечення безпеки програмі потрібна ґрунтовне налаштування.

Avast Premium Security

- Офіційний сайт: avast.com/f-firewall.
- Системні вимоги: Windows XP SP3, Vista, 7, 8, 8.1, 10, 256+ Мбайт ОЗП й 1,5 Гбайт на диску.
- Ліцензія: пробний період на 30 днів.
- Вартість: 1450 гривень за рік, є різні акції. При покупці на два або три роки ціна нижче.
- Мова інтерфейсу: кирилиця.

Хто-небудь не чув про компанію Avast Sofware? Усе про неї чули. Однак, крім відомого антивірусу, Avast випускає ще й файєрвол, який входить у платний набір програм Avast Premium Security (раніше він називався Avast Internet Security, але його перейменували – видиме, щоб уникнути плутанини із програмою, яку ми розглянемо наступною). Тобто окремо завантажити й установити файєрвол не вийде: він іде на застосунок до антивірусу, антиспам-модулю, модулю захисту бездротових мереж і набору інших функцій, платна ліцензія якого коштує 1450 гривень у рік на один ПК.

При установці пакета нам пропонують поставити ще й Google Chrome, але від нього хоча б можна безболісно відмовитися.

Відразу після установки чуда не відбулося: Firewall Tester успішно подолала файєрвол і встановила з'єднання з віддаленим сервером.

Після включення практично всіх можливих налаштувань і параметрів захисту нічого особливо не помінялося. Тільки змусивши файєрвол параноїдально запитувати мене, як реагувати на кожну дію працюючого на комп'ютері програм, я нарешті одержав очікуваний результат.

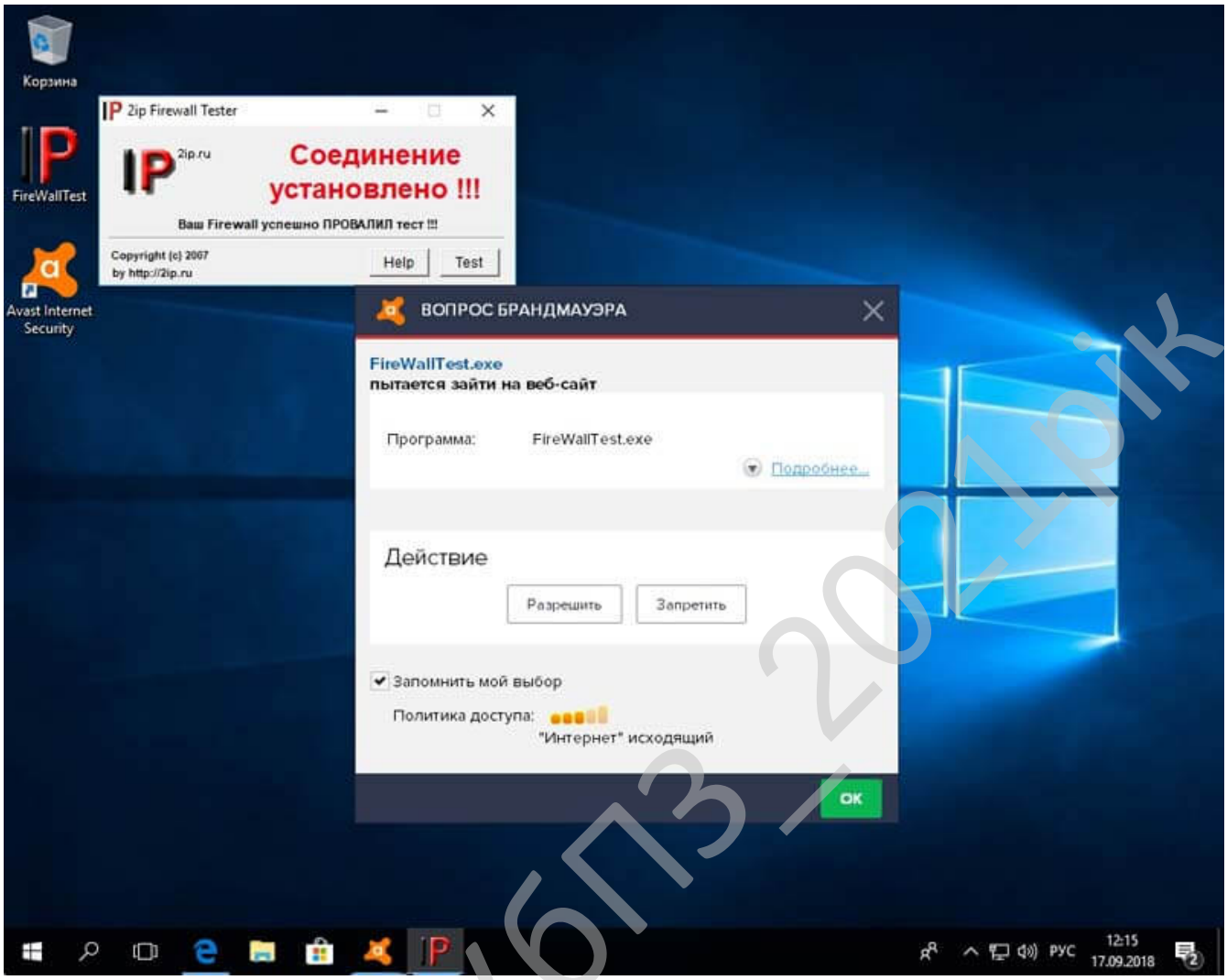

Рисунок 2.2 – Запит дозволу

Налаштування файєрволу розкидані по різних меню, через що знайти з першого разу те, що потрібно, далеко не найпростіше завдання. У комплекті поставки Avast Premium Security є величезна кількість додаткових інструментів для аналізу диска, реєстру, евристики, пошуку вірусів. На жаль немає можливості вилучити весь цей функціонал, щоб залишити тільки один брандмауер. Крім того, пробна версія продукту постійно просить обновитися до версії Pro і заплатити грошей. Мене це Напружує.

AVG Internet Security

- Офіційний сайт: avg.com.
- Системні вимоги: Windows XP SP3, Vista, 7, 8, 8.1, 10.

- Ліцензія: пробний період на 30 днів.
- Вартість: 1990 гривень за рік.
- Мова інтерфейсу: кирилиця.

Безкоштовний антивірус AVG також знаком багатьом, правда про його ефективність існують різні думки. Файєрвол (або, як його називають розроблювачі, «посилений брандмауер») теж не пропонується в якості окремого продукту, а йде в комплекті поставки AVG Internet Security, куди входить ще ціла купа різних програм, включаючи антивірус. Безкоштовної версії ні, але є можливість скачати тріал і протестувати програмне забезпечення безкоштовно протягом місяця.

Примітно, що антивірус AVG не дуже давно був куплений компанією Avast Sofware, але проте продовжує існувати в ролі самостійного продукту. Що ж, давайте подивимося, є чи в ньому якісь істотні відмінності від «материнського» проекту.

Установка AVG починається з пізнаваного віконця інсталятору.

При першому запуску ми бачимо вже знайому картинку.

Можна сказати, що AVG продемонстрував усі те ж саме, що й Avast. І в точності так само зумів розпізнати й заблокувати нашу «шкідливу» програму тільки після примусового включення параноїдального режиму.

За результатами тесту я дійшов висновку, що AVG Internet Security – це по великому рахункові Avast! Premium Security, тільки в профіль тільки під іншою вивіскою. Якщо вам потрібний тільки файєрвол, без антивірусу, антиспама й інших програм, напевно, варто пошукати інший застосунок.

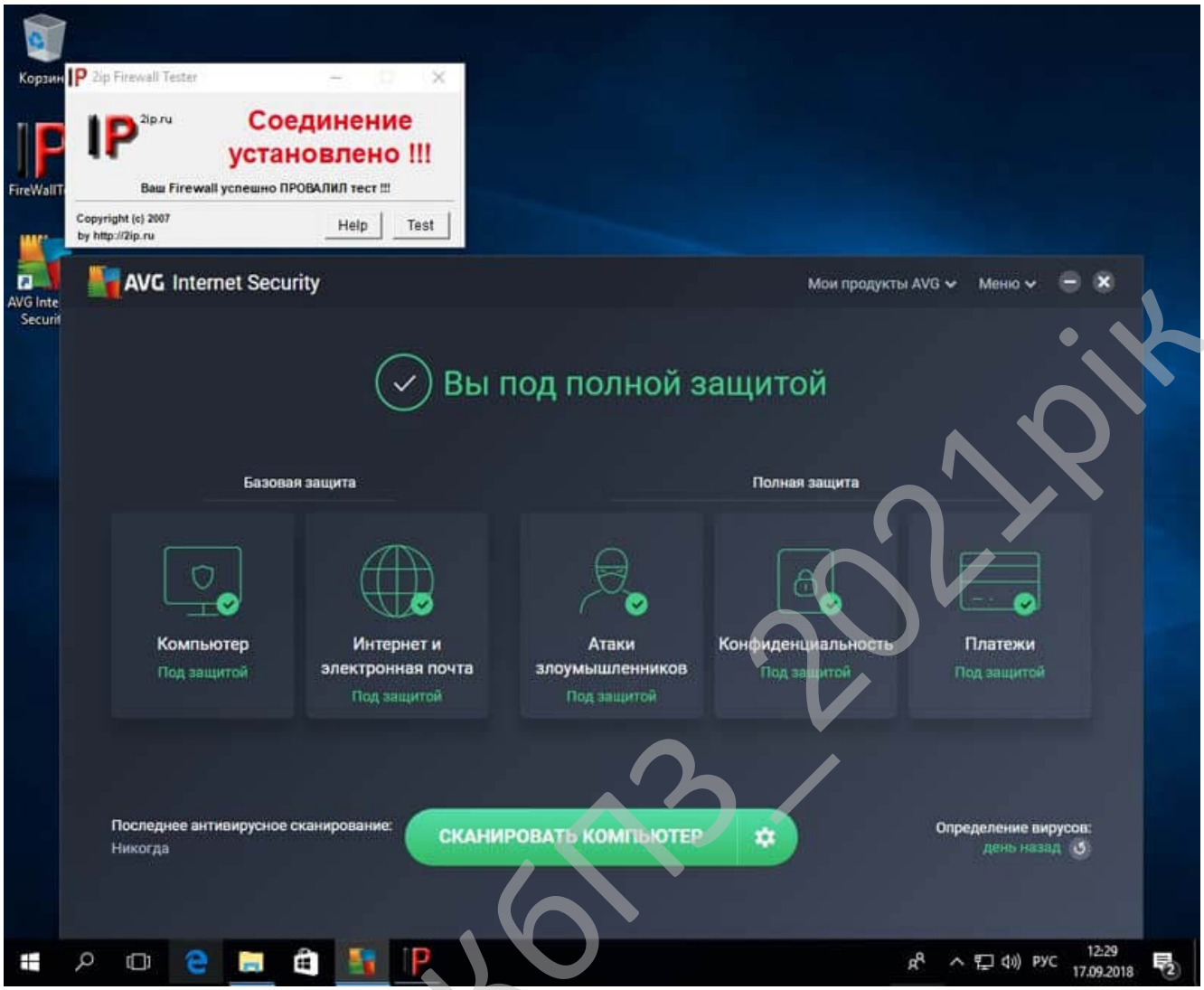

Рисунок 2.3 – Тест AVG Internet Security

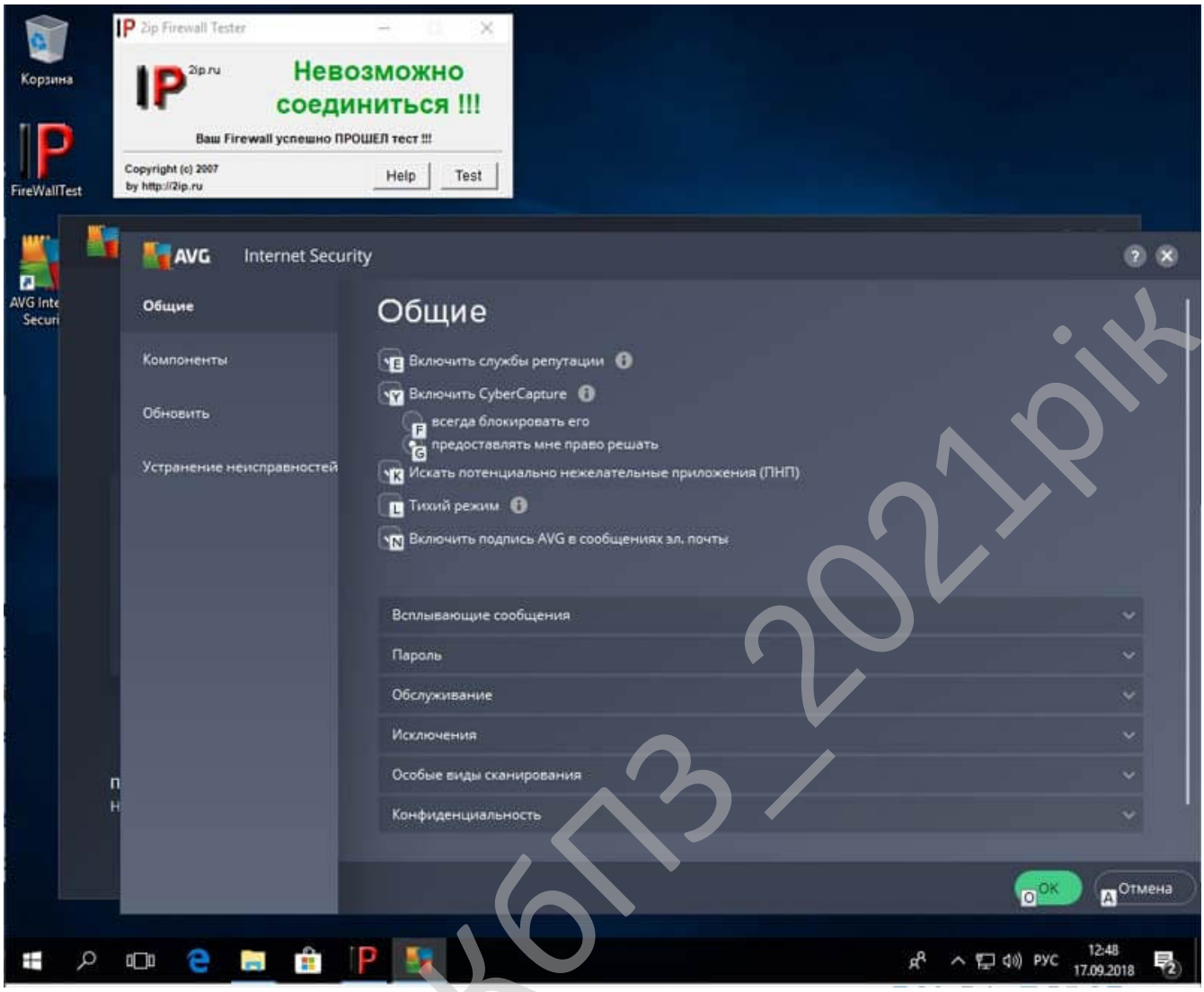

Рисунок 2.4 – Конфігурація файєрволу AVG Internet Security

Outpost Firewall Pro

- Офіційний сайт: відсутній.
- Системні вимоги: Windows XP, Vista, 2003 Server, 2008 Server, 2012 Server, 7, 8, 8.1, 10.
- Ліцензія: пробний період 30 днів.
- Вартість: безцінне.
- Мова інтерфейсу: кирилиця.

У грудні 2015 року Яндекс купив технологію Agnitum, і компанія припинила випускати й підтримувати Outpost Firewall. Але його усе ще можна знайти в інтернеті. Є також безкоштовна версія, але вона, як звичайно, урізана.

Установка програми типова, але при цьому вона хоча б не пропонує в навантаження тонну додаткового й не завжди потрібного ПЗ. Відразу після завершення установки Outpost Firewall Pro просить перезавантажити комп'ютер. Даємо згоду й відразу ж запускаємо Firewall Tester.

Незважаючи на те що Outpost Firewall більше не підтримується, зі своєю основною функцією він справляється на відмінно навіть із налаштуваннями за замовчуванням. А для аматорів покопатися в режимах роботи програми є налаштування правил і можливість включати-відключати інші функції.

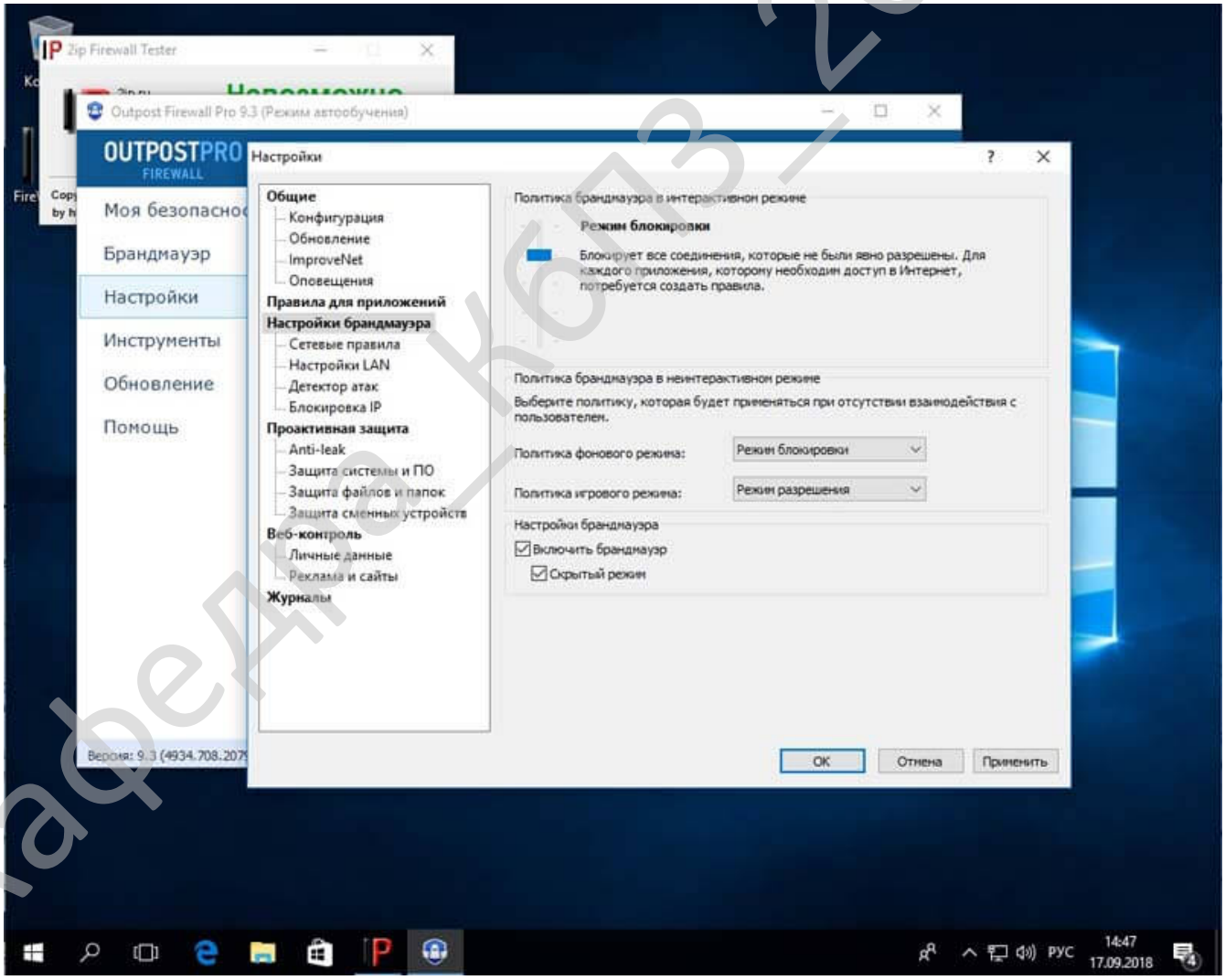

Рисунок 2.5 – Налаштування файєрволу

Програмі ставимо однозначний лайк, жалко тільки, що остання платна версія більше не підтримується. Відновлень теж чекати не доводиться.

Zonealarm Free Firewall

- Офіційний сайт: zonealarm.com.
- Системні вимоги: Windows XP, Vista, 7, 8, 10.
- Ліцензія: Free і Pro.
- Вартість: безкоштовно або 40 доларів у рік.
- Мова інтерфейсу: англійський.

Компанія Zonealarm не так широко відома, як Avast і AVG, але теж захоплюється розробкою антивірусів – у вільне від створення безкоштовного файєрволу час. Zonealarm Firewall має як безкоштовну, так і комерційну версію. Остання відрізняється розширеними налаштуваннями, можливістю включати-відключати компоненти, додавати складні правила фільтрації трафіку й відсутністю настирливої реклами. Увесь інший функціонал продуктів, по заявах розроблювачів, ідентичний.

Після установки й запуску файєрволу Zonealarm пропустив наше з'єднання.

Подивимося, як він себе поведе після невеликого підналаштування.

На жаль, але за допомогою перемикання повзунків і зміни режимів роботи програми Firewall Tester так і не вийшло заблокувати. Тільки після явного блокування це остаточно вдалося. Цікаво, що, коли я копався в налаштуваннях, Zonealarm видав от таке вікно запиту дозволу.

Режим навчання Zonealarm Free Firewall

На скріншоті видне, що в цього файєрволу, крім стандартних варіантів «дозволити», «заборонити» і «навчання», є ще один цікавий режим – kill, який взагалі не дає запуститися застосунку.

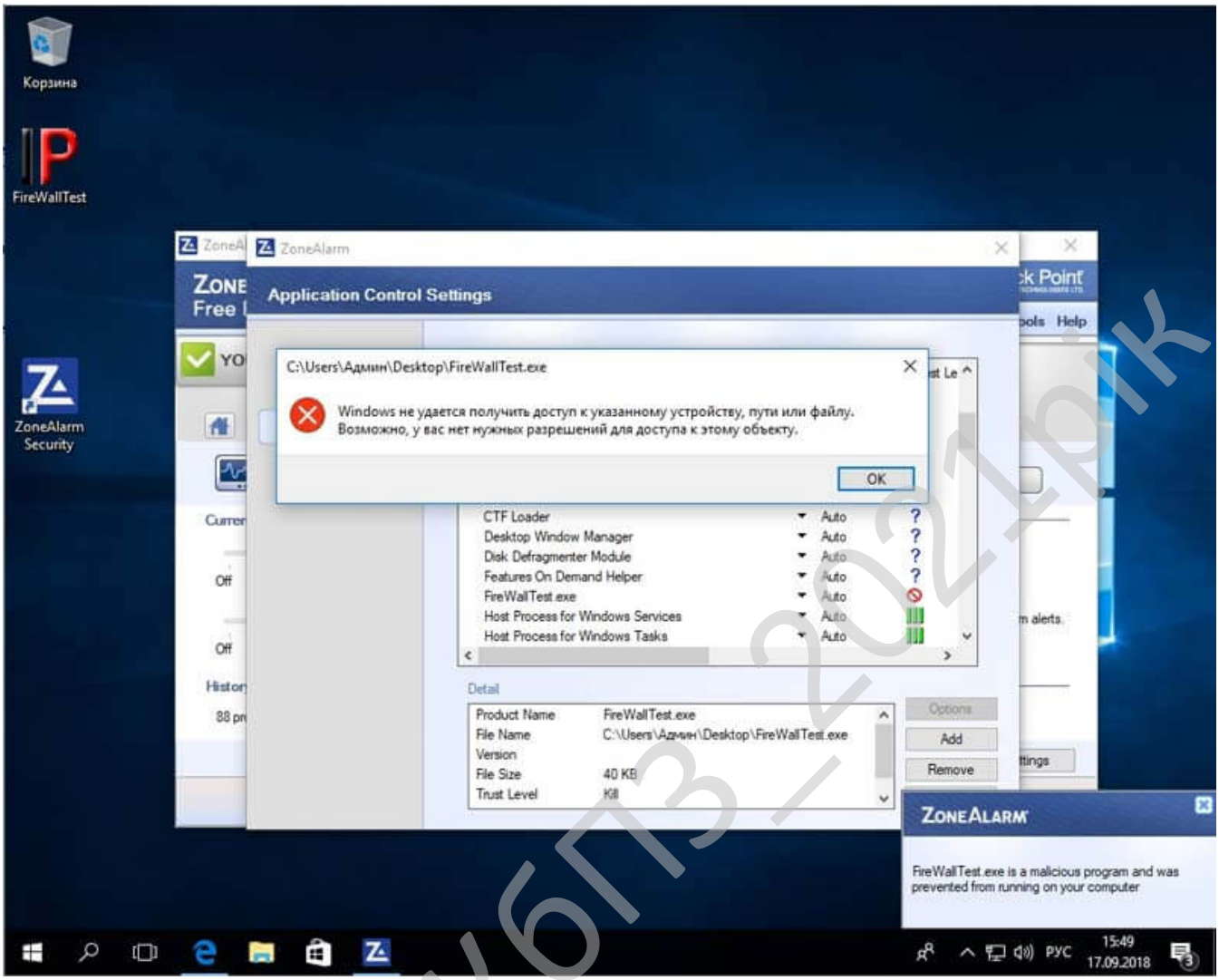

Рисунок 2.6 – Режим kill Zonealarm Free Firewall

Одна проблема: потрібно розуміти, що блокуєш. Простий користувач може через незнання прибити потрібну програму, через що в нього обов'язково відвалиться який-небудь важливий банк-клієнт.

Privatefirewall

- Офіційний сайт: privacyware.com.
- Системні вимоги: Windows XP, Vista, 7, 8, 8.1, 10.
- Ліцензія: Freeware.
- Вартість: безкоштовно.
- Мова інтерфейсу: англійська.

Компанія Privacyware спеціалізується в основному на розробці програмного забезпечення для захисту веб-серверів, тому файєрвол для неї, схоже, побічний продукт життєдіяльності. Можливо, тому Privatefirewall розроблювачем більше не підтримується, і скачати продукт із офіційного сайту зараз не вийде. Зате його дотепер можна знайти на незалежних ресурсах – правда, остання актуальна версія датується аж 2013 роком.

Наш нехитрий тест виявився провалений. Пробуємо ще раз після включення навчального режиму.

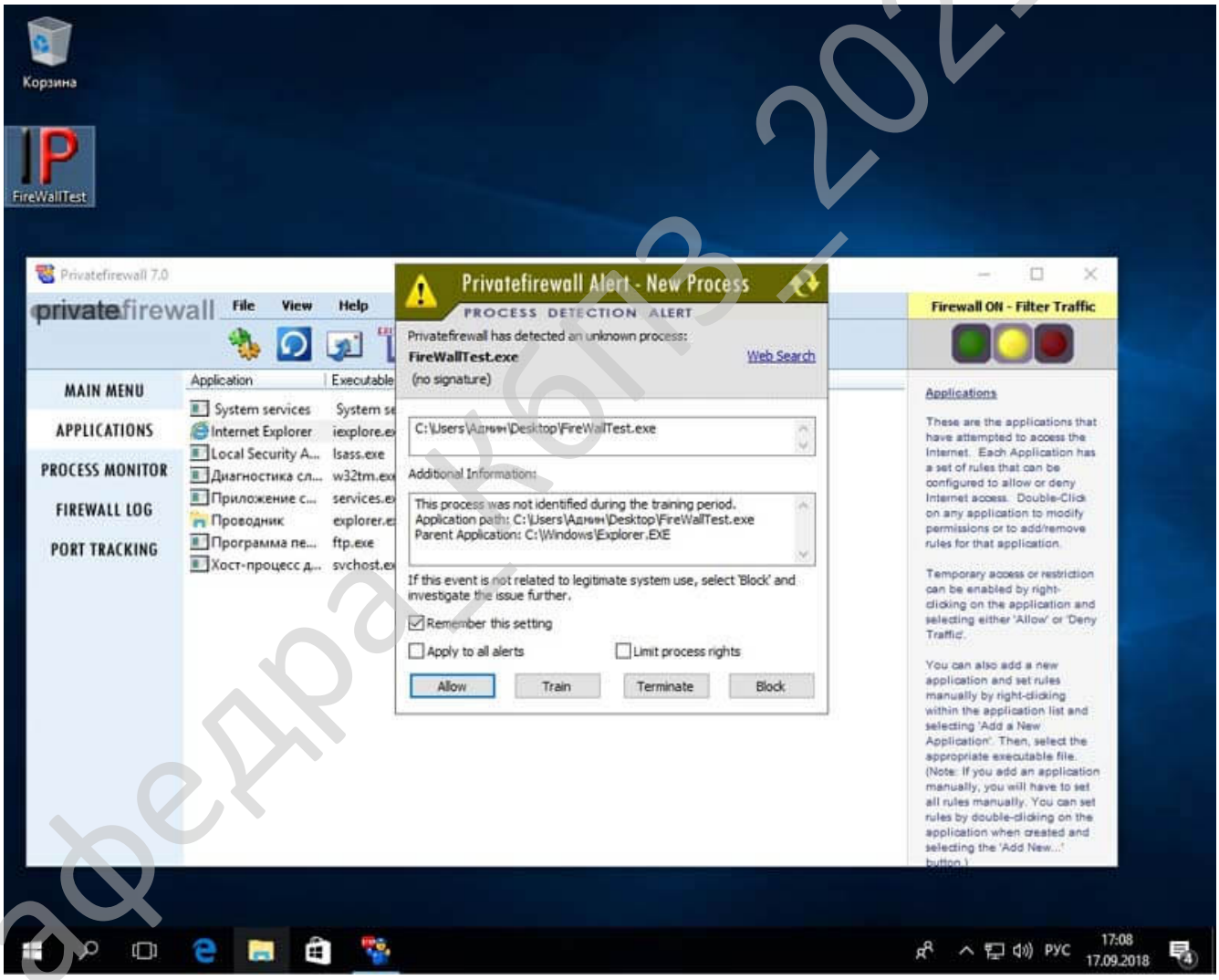

Рисунок 2.7 – Тест Privatefirewall

Цього разу файєрвол справляється із завданням: з'являється вікно із кнопками, що дозволяють заблокувати Firewall Tester. Третій тест в «параноїдальному» режимі програма також проходить успішно, що не дивно.

В Privatefirewall дуже простий інтерфейс, і навіть без знань англійського можна легко в ньому розібратися.

Glasswire

- Офіційний сайт: glasswire.com.
- Системні вимоги: Microsoft Windows 7, 8, 10 (x86,x64), Intel Core 2 Duo or Faster Processor, 4 Гбайт RAM. Мінімальні ще нижче.
- Ліцензія: Shareware.
- Вартість: пробний період сім днів, потім 39 доларів.
- Мова інтерфейсу: кирилиця.

На відміну від попередніх компаній-розроблювачів, для Glasswire файєрвол – це базовий продукт, що, імовірно, повинне позитивно позначитися на якості його роботи. Примітно, що в Glasswire's Firewall є як «настільна», так і мобільна версія, але ми будемо розглядати тільки першу.

Установка програми проста й цілком типова. По її завершенню застосунок покаже нам гарний графік нашої мережевої активності. Однак краса важлива насамперед для творів мистецтва, а нас у першу чергу цікавлять функціональні можливості файєрволу. Запускаємо Firewall Tester.

Незважаючи на те що утиліта успішно з'єдналася з віддаленим сервером, файєрвол відобразив новий процес. Ми можемо його проігнорувати, і він потрапить у список довірених, а можемо відразу заблокувати. Виставивши більш жорсткі налаштування політики дозволів, ми бачимо, що наш процес відразу викликає підозри файєрволу й успішно блокується.

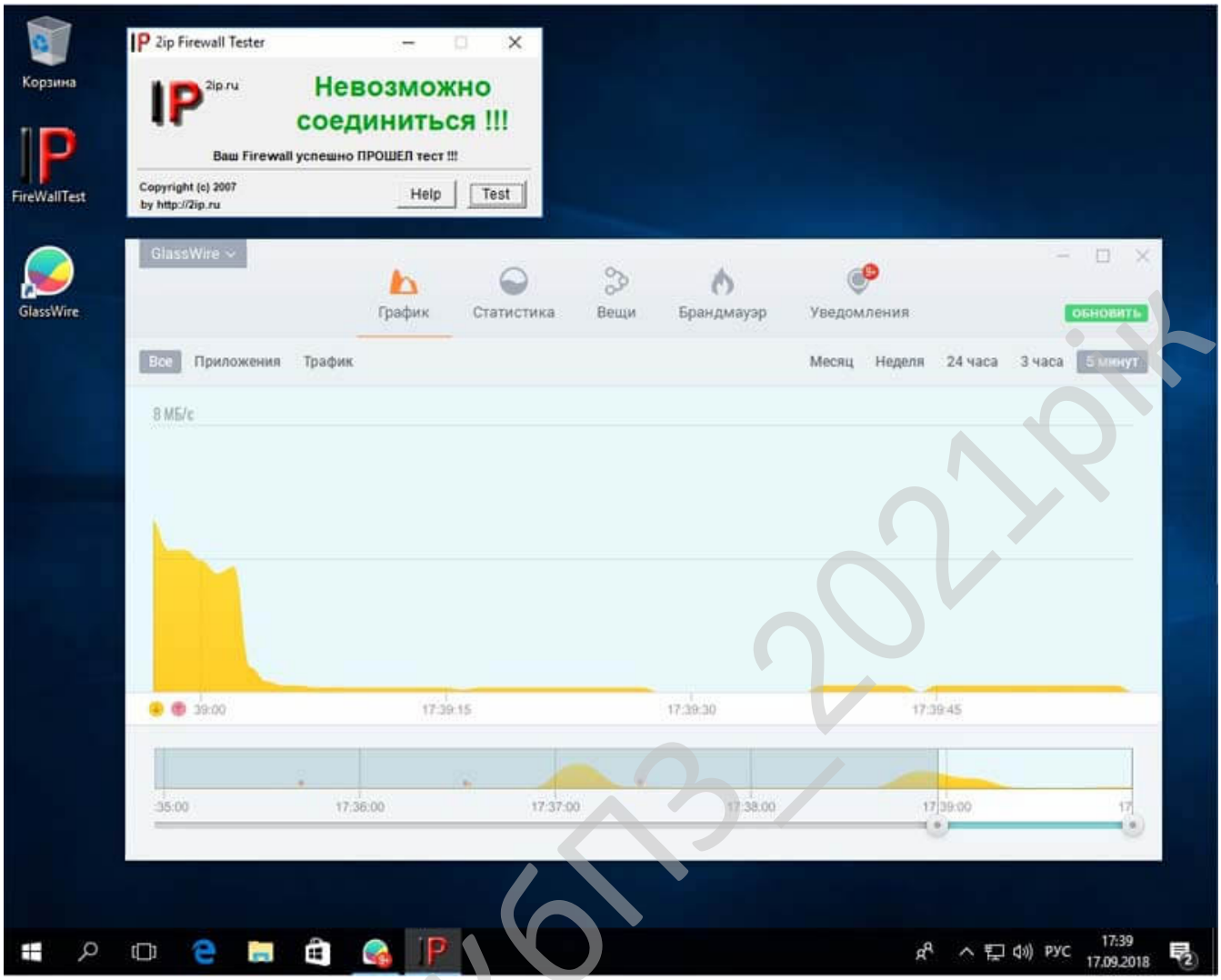

Рисунок 2.8 – Тест Privatefirewall

Ця програма сподобалася мені більше інших розглянутих файєрволів, єдиний її великий недолік – платна ліцензія, сама бюджетна версія якої коштує 39 доларів.

Tinywall

- Офіційний сайт: tinywall.pados.hu.
- Системні вимоги: Microsoft Windows 7, 8, 10 (x86,x64), Intel Core 2 Duo or Faster Processor, 4 Гбайт RAM. Мінімальні ще нижче.
- Ліцензія: Freeware.
- Вартість: безкоштовно.
- Мова інтерфейсу: кирилиця.

Ця невеликий безкоштовний файєрвол – результат праць угорського розроблювача Кароя Падоша (Károly Pados). Утиліта цікава тим, що працює із трея – трохи незвично. Але при цьому вужі при проведенні першого тесту не дає пробитися в мережу нашому «вредоносу».

Саме забавне, що при включенні навчального режиму файєрвол пропускає весь трафік і, по ідеї, повинен на підставі діалогів або внутрішнього механізму вирішувати, що робити із запущеними процесами. У цьому випадку вибір виявився невірний.

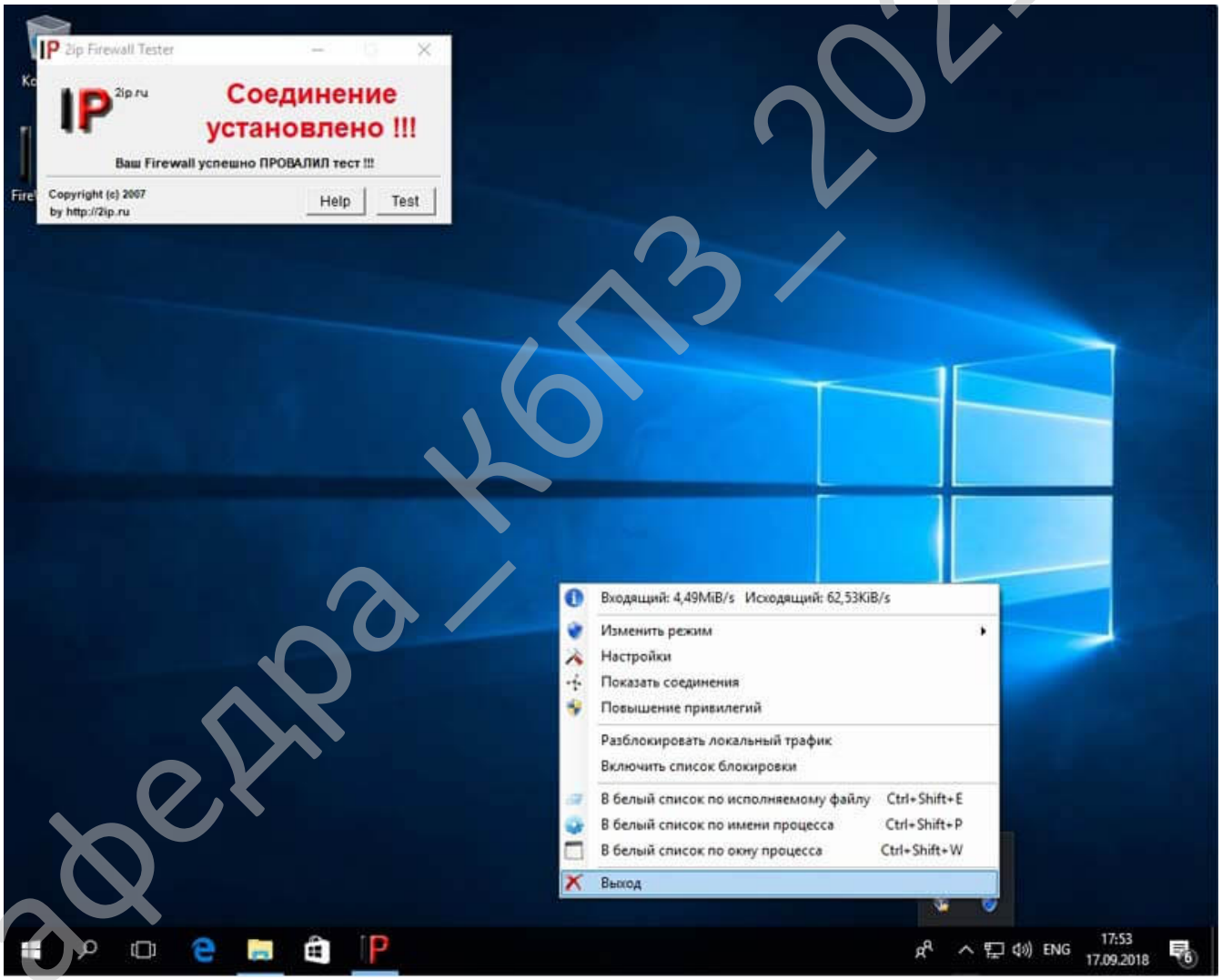

Рисунок 2.9 – Тест Tinywall

Однак при складанні білого списку Tinywall очікуване впорався з поставленим завданням.

Виводи: перед нами дуже компактний і «полегшений» файєрвол з невеликою кількістю налаштувань. Однак цього самого налаштування він однаково вимагає: у варіанті з коробки програма мишей не ловить.

Якщо дивитися на вибір файєрволу очима звичайного користувача, то найбільш удалим варіантом буде Tinywall або дідок Outpost Firewall Pro. Незважаючи на гарні показники Glasswire, рекомендувати цю програму я не стану через високу вартість. У принципі, можна використовувати будь-який розглянутий у статті файєрвол. Головне – це правильне налаштування.

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 варіанту 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 кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки DMZ з використанням технології Reverse Access.

В процесі розробки кваліфікаційної бакалаврської роботи 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 кібербезпеки, 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

Дана робота містить огляд п'яти варіантів застосунку завдання організації доступу до сервісів корпоративної мережі з Інтернет. У рамках огляду приводиться аналіз варіантів на предмет безпеки й реалізуємості, що допоможе розібратися в суті питання, освіжити й систематизувати свої знання як починаючим фахівцям, так і більш досвідченим. Матеріали роботи можна використовувати для обґрунтування проектних застосунків.

При розгляді варіантів як прикладу візьмемо мережу, у якій потрібно опублікувати:

- Корпоративний поштовий сервер (WEB-mail).
- Корпоративний термінальний сервер (RDP).
- Extranet сервіс для контрагентів (WEB-api).

Варіант 1. Плоска мережа

У даному варіанті всі вузли корпоративної мережі втримуються в одній, загальної для всіх мережі («Внутрішня мережа»), у рамках якої комунікації між ними не обмежуються. Мережа підключена до Інтернет через прикордонний маршрутизатор/міжмережевий екран (далі – IFW).

Доступ вузлів в Інтернет здійснюється через NAT, а доступ до сервісів з Інтернет через Port forwarding.

Переваги варіанту:

- Мінімальні вимоги до функціонала IFW (можна зробити практично на будь-якому, навіть домашньому роутері).
- Мінімальні вимоги до знань фахівця, що здійснює реалізацію варіанту.

Недоліки варіанту:

– Мінімальний рівень безпеки. У випадку злому, при якому Порушник одержить контроль над одним з опублікованих в Інтернеті серверів, йому для подальшої атаки стають доступні всі інші вузли й канали зв'язки корпоративної мережі.

Аналогія з реальним життям

Подібну мережу можна зрівняти з компанією, де персонал і клієнти перебувають в одній спільній кімнаті (open space)

Варіант 2. DMZ

Для усунення зазначеного раніше недоліку вузли мережі, доступні з Інтернет, поміщають у спеціально виділений сегмент – демілітаризовану зону (DMZ). DMZ організує за допомогою міжмережевих екранів, що відокремлюють її від Інтернет (IFW) і від внутрішньої мережі (DFW).

При цьому правила фільтрації міжмережевих екранів виглядають у такий спосіб:

– Із внутрішньої мережі можна ініціювати з'єднання в DMZ і в WAN (Wide Area Network).

– З DMZ можна ініціювати з'єднання в WAN.

– З WAN можна ініціювати з'єднання в DMZ.

– Ініціація з'єднань із WAN і DMZ до внутрішньої мережі заборонена.

Переваги варіанту:

– Підвищена захищеність мережі від зломів окремих сервісів. Навіть якщо один із серверів буде зламаний, Порушник не зможе одержати доступ до ресурсів, що перебувають у внутрішній мережі (наприклад, мережевим принтерам, системам відеоспостереження і т.д.).

Недоліки варіанту:

– Сам по собі винос серверів в DMZ не підвищує їхня захищеність.

– Необхідний додатковий файєрвол для відділення DMZ від внутрішньої мережі.

Аналогія з реальним життям

Даний варіант архітектури мережі схожий на організацію робочої й клієнтської зон у компанії, де клієнти можуть перебувати тільки в клієнтській зоні, а персонал може бути як у клієнтській, так і в робочих зонах. DMZ сегмент – це саме і є аналог клієнтської зони.

Варіант 3. Поділ сервісів на Front-End і Back-End

Як ми вже відзначали раніше, розміщення сервера в DMZ жодним чином не поліпшує безпека самого сервісу. Одним з варіантів виправлення ситуації є поділ функціонала сервісу на дві частини: Front-End і Back-End. При цьому кожна частина розташовується на окремому сервері, між якими організує мережеву взаємодію. Сервера Front-End, що реалізують функціонал взаємодії із клієнтами, що перебувають в Інтернет, розміщають в DMZ, а сервера Back-End, що реалізують інший функціонал, залишають у внутрішній мережі. Для взаємодії між ними на DFW створюють правила, що дозволяють ініціацію підключень від Front-End до Back-End.

Як приклад розглянемо корпоративний поштовий сервіс, що обслуговує клієнтів як зсередини мережі, так і з Інтернет. Клієнти зсередини використовують POP3/SMTP, а клієнти з Інтернет працюють через WEB-інтерфейс. Звичайно на етапі впровадження компанії вибирають найбільш простий спосіб розгортання сервісу й ставлять усі його компоненти на один сервер. Потім, у міру усвідомлення необхідності забезпечення інформаційної безпеки, функціонал сервісу розділяють на частині, і та частина, що відповідає за обслуговування клієнтів з Інтернет (Front-End), виноситься на окремий сервер, який по мережі взаємодіє із сервером, що реалізують функціонал, що залишився (Back-End). При цьому Front-End розміщають в DMZ, а Back-End залишається у внутрішньому сегменті. Для зв'язку між Front-End і Back-End на DFW створюють правило, що дозволяє, ініціацію з'єднань від Front-End до Back-End.

Переваги варіанту:

– У загальному випадку атаки, спрямовані проти сервісу, що захищається, можуть «спіткнутися» про Front-End, що дозволить нейтралізувати або суттєво знизити можливий збиток. Наприклад, атаки типу TCP SYN Flood або slow http read, спрямовані на сервіс, приведуть до того, що Front-End сервер може виявитися недоступний, у той час як Back-End буде продовжувати нормально функціонувати й обслуговувати користувачів.

– У загальному випадку на Back-End сервері може не бути доступу в Інтернет, що у випадку його злому (наприклад, локально запущеним шкідливим кодом) утруднить віддалене керування їм з Інтернет.

– Front-End добре підходить для розміщення на ньому міжмережевого екрана рівня застосунків (наприклад, WEB application firewall) або системи запобігання вторгнень (IPS, наприклад snort).

Недоліки варіанту:

– Для зв'язку між Front-End і Back-End на DFW створюється правило, що дозволяє ініціацію з'єднання з DMZ у внутрішню мережу, що породжує погрози, пов'язані з використанням даного правила з боку інших вузлів в DMZ (наприклад, за рахунок реалізації атак IP spoofing, ARP poisoning і т.д.).

– Не всі сервіси можуть бути розділені на Front-End і Back-End.

– У компанії повинні бути реалізовані бізнес-процеси актуалізації правил міжмережевого екранування.

– У компанії повинні бути реалізовані механізми захисту від атак з боку Порушників, що одержали доступ до сервера в DMZ.

Примітки:

– У реальному житті навіть без поділу серверів на Front-End і Back-End серверам з DMZ дуже часто необхідно звертатися до серверів, що перебувають у внутрішній мережі, тому зазначені недоліки даного варіанту будуть також слушні й для попереднього розглянутого варіанту.

– Якщо розглядати захист застосунків, що працюють через WEB-інтерфейс, то навіть якщо сервер не підтримує рознесення функцій на Front-End і Back-End, застосування http reverse proxy сервера (наприклад, nginx) у якості Front-End дозволить мінімізувати ризики, пов'язані з атаками на відмову в обслуговуванні. Наприклад, атаки типу SYN flood можуть зробити http reverse proxy недоступним, у той час як Back-End буде продовжувати працювати.

Аналогія з реальним життям

Даний варіант по суті схожий на організацію праці, при якій для високо завантажених працівників використовують помічників – секретарів. Тоді Back-End буде аналогом завантаженого працівника, а Front-End аналогом секретаря.

Варіант 4. Захищений DMZ

DMZ це частина мережі, доступна з Internet, і, як наслідок, піддана максимальному ризику компрометації вузлів. Дизайн DMZ і застосовувані в ній підходи повинні забезпечувати максимальну живучість в умовах, коли Порушник одержав контроль над одним з вузлів в DMZ.

Переваги варіанту:

– Високий ступінь безпеки.

Недоліки варіанту:

– Підвищені вимоги до функціональних можливостей устаткування.

– Працезатрати у впровадженні й підтримці.

Аналогія з реальним життям

Якщо раніше DMZ ми зрівняли із клієнтською зоною, оснащеної диванчиками й пуфиками, то захищений DMZ буде більше схожий на броньовану касу.

Варіант 5. Back connect

Розглянуті в попередньому варіанті заходу захисти були засновані на тому, що в мережі був присутній пристрій (комутатор / маршрутизатор / міжмережевий екран), здатне їх реалізовувати. Але на практиці, наприклад, при

використанні віртуальної інфраструктури (віртуальні комутатори найчастіше мають дуже обмежені можливості), подібного пристрою може й не бути.

У цих умовах Порушникові стають доступні багато з розглянутих раніше атак, найнебезпечнішими з яких будуть:

– атаки, що дозволяють перехоплювати й модифікувати трафік (ARP Poisoning, CAM table overflow + TCP session hijacking і ін.);.

– атаки, пов'язані з експлуатацією уразливостей серверів внутрішньої мережі, до яких можна ініціювати підключення з DMZ (що можливо шляхом обходу правил фільтрації DFW за рахунок IP і MAC spoofing).

Наступної немаловажною особливістю, яку ми раніше не розглядали, але яка не перестає бути від цього менш важливої, це те, що автоматизовані робочі місця (АРМ) користувачів теж можуть бути джерелом (наприклад, при зараженні вірусами або троянами) шкідливого впливу на сервера.

Таким чином, перед нами встає завдання захистити сервера внутрішньої мережі від атак Порушника як з DMZ, так і із внутрішньої мережі (зараження АРМа трояном можна інтерпретувати як дії Порушника із внутрішньої мережі).

Пропонований далі підхід спрямований на зменшення числа каналів, через які Порушник може атакувати сервера, а таких каналу як мінімум два. Перший це правило на DFW, що дозволяє доступ до сервера внутрішньої мережі з DMZ (нехай навіть і з обмеженням по IP-адресам), а другий – це відкритий на сервері мережевий порт, по якому очікуються запити на підключення.

Закрити зазначені канали можна, якщо сервер внутрішньої мережі буде сам будувати з'єднання до сервера в DMZ і буде робити це за допомогою криптографічно захищених мережевих протоколів. Тоді не буде ні відкритого порту, ні правила на DFW.

Але проблема в тому, що звичайні серверні служби не вміють працювати подібним чином, і для реалізації зазначеного підходу необхідно застосовувати мережеве туннелювання, реалізоване, наприклад, за допомогою SSH або VPN, а

вже в рамках тунелів дозволяти підключення від сервера в DMZ до сервера внутрішньої мережі.

Загальна схема роботи даного варіанту виглядає в такий спосіб:

– На сервер в DMZ інсталюється SSH/VPN сервер, а на сервер у внутрішній мережі інсталюється SSH/VPN клієнт.

– Сервер внутрішньої мережі ініціює побудова мережевого тунелю до сервера в DMZ. Тунель будується із взаємною автентифікацією клієнта й сервера.

– Сервер з DMZ у рамках побудованого тунелю ініціює з'єднання до сервера у внутрішній мережі, по якому передаються дані, що захищаються.

– На сервері внутрішньої мережі настроюється локальний міжмережевий екран, що фільтрує трафік, що проходить по тунелю.

Використання даного варіанту на практиці показало, що мережеві тунелі зручно будувати за допомогою Openvpn, оскільки він має наступні важливі властивості:

– Кроссплатформенність. Можна організовувати зв'язок на серверах з різними операційними системами.

– Можливість побудови тунелів із взаємної автентифікацією клієнта й сервера.

– Можливість використання сертифікованої криптографії.

На перший погляд може здатися, що дана схема зайво ускладнена й що, раз на сервері внутрішньої мережі однаково потрібно встановлювати локальний міжмережевий екран, то простіше зробити, щоб сервер з DMZ, як звичайно, сам підключався до сервера внутрішньої мережі, але робив це по шифрованому з'єднанню. Дійсно, даний варіант закриє багато проблем, але він не зможе забезпечити головного – захист від атак на уразливості сервера внутрішньої мережі, чинених за рахунок обходу міжмережевого екрана за допомогою IP і MAC spoofing.

Переваги варіанту:

– Архітектурне зменшення кількості векторів атак на, що захищається сервер внутрішньої мережі.

– Забезпечення безпеки в умовах відсутності фільтрації мережевого трафіку.

– Захист даних, переданих по мережі, від несанкціонованого перегляду й зміни.

– Можливість вибіркового підвищення рівня безпеки сервісів.

– Можливість реалізації двоконтурної системи захисту, де перший контур забезпечується за допомогою міжмережевого екранування, а другий організує на базі даного варіанту.

Недоліки варіанту:

– Впровадження й супровід даного варіанту захисту вимагає додаткових трудових витрат.

– Несумісність із мережевими системами виявлення й запобігання вторгнень (IDS/IPS).

– Додаткове обчислювальне навантаження на сервера.

Аналогія з реальним життям

Основний зміст даного варіанту в тому, що довірена особа встановлює зв'язок з не довіреним, що схоже на ситуацію, коли при видачі кредитів Банки самі передзвонюють потенційному позичальникові з метою перевірки даних.

Отже, ми розглянули всі п'ять заявлених варіантів організації доступу до сервісів корпоративної мережі з Інтернет. Який з них краще, який гірше – сказати складно, оскільки все залежить, в остаточному підсумку, від тієї інформації, яку необхідно захистити, і тих ресурсів, якими компанія розташовує для захисту. Якщо ні ресурсів, ні знань ні, то оптимальним буде перший варіант. Якщо ж інформація дуже коштовна, то комбінація четвертого й п'ятого варіантів дасть неперевершений рівень безпеки.

3.2 Розробка структурної схеми

В еру хмарних обчислень DMZ (Demilitarized Zone, демілітаризована зона, DMZ – фізичний або логічний сегмент мережі, що містить, що й надає організації загальнодоступні сервіси, а також віддільний їх від інших ділянок локальної мережі, що дозволяє забезпечити внутрішньому інформаційному простору додатковий захист від зовнішніх атак) стала набагато більш важливої – і одночасно більш уразливої, – чому коли-або могли собі це представити її первісні архітектори.

Десять або двадцять років тому, коли більшість кінцевих точок усе ще перебували у внутрішній мережі компанії, DMZ зона була всього лише додатковою прибудовою до мережі. Користувачі, що перебувають за межами локальної обчислювальної мережі, дуже рідко запитували доступ до внутрішніх сервісів, і навпаки, необхідність у доступі локальних користувачів до загальнодоступних сервісів також виникала не часто, тому в той час DMZ мало що вирішувала в справі забезпечення інформаційної безпеки. Що ж стало з її роллю зараз?

Сьогодні ви або софтверна компанія, або ви працюєте з величезною кількістю постачальників SaaS-сервісів (Software as a service, програмне забезпечення як послуга). Так чи інакше, але вам постійно доводиться надавати доступ користувачам, що перебувають за межами вашої LAN, або запитувати доступ до сервісів, розташованих у хмарі. У результаті ваша DMZ «забита під зав'язку» різними додатками. І хоча DMZ, як споконвічно передбачалося, повинна служити своєрідною контрольною точкою вашого периметра, у наші дні її функція усе більше нагадує зовнішню рекламну вивіску для кіберзлочинців.

Кожний сервіс, який ви запускаєте в DMZ зоні, є ще одним інформаційним повідомленням для потенційних хакерів про те, скільки у вас користувачів, де ви зберігаєте свою критично важливу ділову інформацію, і чи містять ці дані те, що зловмисник може захотіти украсти. Нижче представлено

чотири кращі практики, які дозволять вам включити й настроїти DMZ так, що припинити весь цей бардак.

1. Зробіть DMZ дійсно окремим сегментом мережі

Основна ідея, що лежить в основі DMZ, полягає в тому, що вона повинна бути дійсно відділена від інший LAN. Тому вам необхідно включити одмінні між собою політики IP-маршрутизації й безпеки для DMZ і іншої частини вашої внутрішньої мережі. Це на порядок ускладнить життя нападаючим на вас кіберзлочинцям, адже навіть якщо вони розберуться з вашою DMZ, ці знання вони не зможуть використовувати для атаки на вашу LAN.

2. Настройте сервіси усередині й поза DMZ зони

В ідеалі всі сервіси, які перебувають за межами вашої DMZ, повинні встановлювати пряме підключення тільки до самої DMZ зоні. Сервіси, розташовані усередині DMZ, повинні підключатися до зовнішнього світу тільки через проксі-сервери. Сервіси, що перебувають усередині DMZ, більш безпечні, чому розташовані поза неї. Сервіси, які краще захищені, повинні побрати на себе роль клієнта при здійсненні запиту з менш захищених областей.

3. Використовуйте два міжмережеві екрани для доступу до DMZ

Хоча можна включити DMZ, використовуючи тільки один файєрвол із трьома або більш мережевими інтерфейсами, налаштування, що використовує два міжмережеві екрани надасть вам більш надійні засоби стримування кіберзлочинців. Перший брандмауер використовується на зовнішньому периметрі й служить тільки для напрямку трафіку виняткова на DMZ. Другий – внутрішній міжмережевий екран – обслуговує трафік з DMZ у внутрішню мережу. Такий підхід уважається більш безпечним, тому що він створює дві окремі незалежні перешкоди на шляху хакера, що застосунав атакувати вашу мережу.

4. Впровадьте технологію Reverse Access

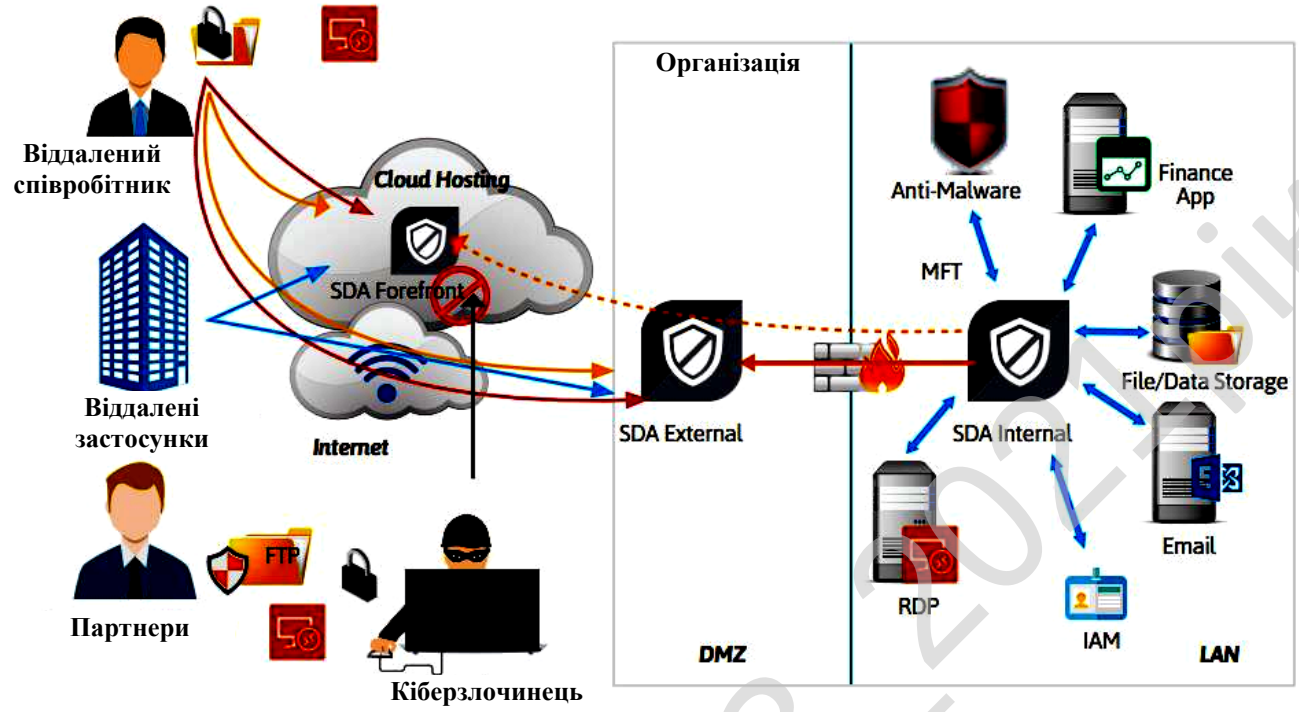

Рисунок 3.1 – Структурна схема системи

Технологія Reverse Access – зробить DMZ ще більш безпечною. Ця технологія, заснована на використанні двох серверів, усуває необхідність відкриття будь-яких портів у міжмережевому екрані, у те ж саме час забезпечуючи безпечний доступ до застосунків між мережами (через файєрвол). Застосунок містить у собі:

– Зовнішній сервер – установлюється в DMZ / зовнішньому / незахищеному сегменті мережі.

– Внутрішній сервер – установлюється у внутрішньому / захищеному сегменті мережі.

Роль зовнішнього сервера, розташованого в DMZ організації (на місці або в хмарі), полягає в підтримці клієнтської сторони користувацького інтерфейсу (фронтенда, front-end) до різних сервісів і додаткам, що перебувають у Всесвітній мережі. Він функціонує без необхідності відкриття яких-небудь портів у

внутрішньому брандмауерові й гарантує, що у внутрішню локальну мережу можуть потрапити тільки легітимні дані сеансу. Зовнішній сервер виконує розвантаження TCP, дозволяючи підтримувати роботу з будь-яким додатком на основі TCP без необхідності розшифровувати дані трафіку криптографічного протоколу SSL (Secure Sockets Layer, рівень захищених сокетів).

Роль внутрішнього сервера полягає в тому, щоб провести дані сеансу у внутрішню мережу із зовнішнього вузла SDA (Software-Defined Access, програмно-обумовлений доступ), і, якщо тільки сеанс є легітимним, виконати функціональність проксі-сервера рівня 7 (розвантаження SSL, переписування Url-адрес, DPI (Deep Packet Inspection, докладний аналіз пакетів) і т.д.) і пропустити його на адресований сервер застосунків.

Технологія Reverse Access дозволяє автентифікувати дозвіл на доступ користувачам ще до того, як вони зможуть одержати доступ до ваших критично-важливих застосунків. Зловмисник, що одержав доступ до ваших застосунків через незаконний сеанс, може досліджувати вашу мережу, намагатися проводити атаки ін'єкції коду або навіть пересуватися по вашій мережі. Але позбавлений можливості позиціонувати свою сесію як легітимн зловмисник, що атакує вас, втрачає більшої частини свого інструментарію, стаючи значно більш обмеженим у засобах.

3.3 Розробка функціональної схеми

У якості можливих атак розглянемо атаки, яким піддані практично всі інформаційні системи, що працюють із налаштуваннями за замовчуванням:

1. Cam-table overflow.
2. ARP poisoning.
3. Rogue DHCP Server.
4. DHCP starvation.
5. VLAN hopping.

6. MAC flood.
7. UDP flood.
8. TCP SYN flood.
9. TCP session hijacking.
10. TCP reset.
11. Атаки на WEB-застосунки.
12. Атаки на обхід засобів автентифікації й авторизацію від імені легітимного користувача (наприклад, добір паролів, PSK і т.д.).
13. Атаки на уразливості в мережевих службах, наприклад:.
14. Атака на WEB-сервер – slow reading.
15. DNS cache poisoning.

Більша частина зазначених атак (принаймні з 1 по 10) базується на уразливостях архітектури сучасних Ethernet/IP мереж, що полягають у можливості Порушника підробляти в мережевих пакетах MAC і IP адреси. Експлуатацію даних уразливостей іноді виділяють в окремий види атак:

- MAC spoofing;.
- IP spoofing.

Тому побудова системи захисту DMZ почнемо з розгляду способів захисту від IP і MAC spoofing.

Наведені нижче способи захисту від даних атак не є єдино можливими. Існують і інші способи.

Захист від MAC spoofing

Нейтралізацією даної атаки може бути фільтрація Mac-адрес на портах комутатора. Наприклад, трафік по портові 3 повинен проходити тільки у випадку, якщо в адресі джерела або в адресі призначення зазначений Mac-адреса DE:AD:BE:AF:DE:AD або широкомовна адреса (у деяких випадках).

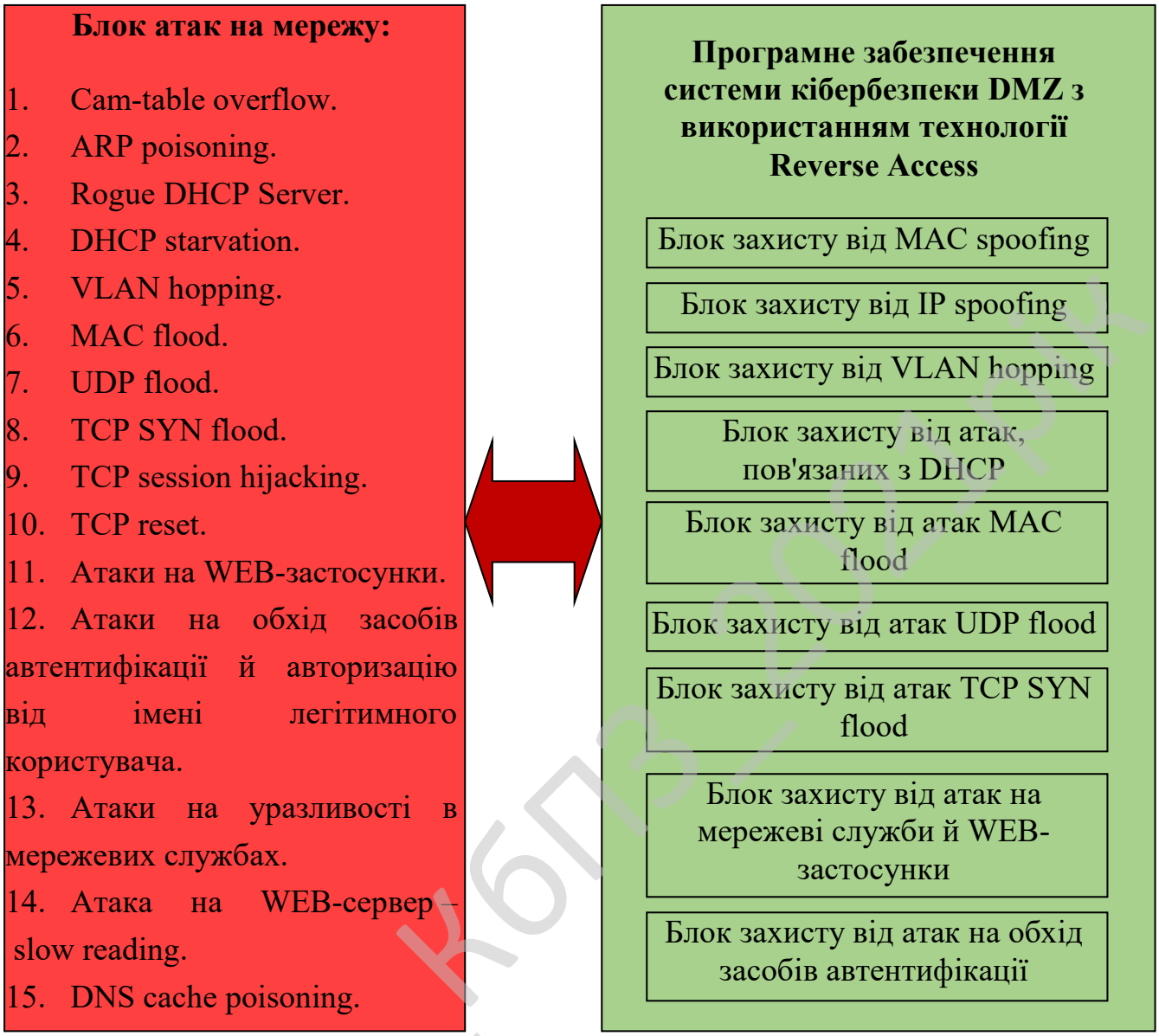

Рисунок 3.2 – Функціональна схема системи

Захист від IP spoofing

Схема атаки IP spoofing схожа на попередню, за винятком того, що Порушник підробляє не MAC, а IP-адресу. Захист від IP spoofing може бути реалізована шляхом поділу IP-мережі DMZ на більш дрібні IP-підмереж й подальшою фільтрацією трафіку на інтерфейсах маршрутизатора за аналогією з розглянутої раніше Mac-фільтрацією. Нижче приклад дизайну DMZ, що реалізує даний принцип.

В DMZ розташовується 3 вузла:

- Термінальний сервер (192.168.100.2).
- Поштовий сервер (192.168.100.5).
- Extranet сервер (192.168.100.9).

Для DMZ виділена IP-мережа 192.168.100.0/24, у даній мережі виділяються 3 IP-підмереж (по числу серверів):

Підмережа 1 – 192.168.100.0/30 для термінального сервера (192.168.100.2)

Підмережа 2 – 192.168.100.4/30 для поштового сервера (192.168.100.5)

Підмережа 3 – 192.168.100.8/30 для поштового сервера (192.168.100.9)

На практиці поділ мережі на подібні підмереж реалізують за допомогою технології VLAN. Однак, її застосування породжує ризики, захист від яких ми зараз розглянемо.

Захист від VLAN hopping

Для захисту від цієї атаки на комутаторі відключають можливість автоматичного узгодження типів (trunk / access) портів, а самі типи адміністратор призначає вручну. Крім того, організаційними заходами забороняється використання так званого native VLAN.

Захист від атак, пов'язаних з DHCP

Незважаючи на те, що DHCP призначений для автоматизації конфігурування IP-адрес робочих станцій, у деяких компаніях зустрічаються випадки, коли через DHCP видаються IP-адреси для серверів, але це досить погана практика. Тому для захисту від Rogue DHCP Server, DHCP starvation рекомендується повна відмова від DHCP в DMZ.

Захист від атак MAC flood

Для захисту від MAC flood проводять налаштування на портах комутатора на предмет обмеження граничної інтенсивності широкомовного трафіку (оскільки звичайно при даних атаках генерується широкомовний трафік (broadcast)). Атаки, пов'язані з використанням конкретних (unicast) мережевих адрес, будуть заблоковані MAC фільтрацією, яку ми розглянули раніше.

Захист від атак UDP flood

Захист від даного типу атак проводиться аналогічно захисту від MAC flood, за винятком того, що фільтрація здійснюється на рівні IP (L3).

Захист від атак TCP SYN flood

Для захисту від даної атаки можливі варіанти:

– Захист на вузлі мережі за допомогою технології TCP SYN Cookie.

– Захист на рівні міжмережевого екрана (за умови поділу DMZ на підмереж) шляхом обмеження інтенсивності трафіку, що містить запити TCP SYN.

Захист від атак на мережеві служби й WEB-застосунки

Універсального рішення даної проблеми ні, але устояною практикою є впровадження процесів керування уразливостями ПЗ (виявлення, установка патчів і т.д., наприклад, так), а також використання систем виявлення й запобігання вторгнень (IDS/IPS).

Захист від атак на обхід засобів автентифікації

Як і для попереднього випадку універсального рішення даної проблеми немає.

Звичайно у випадку великої кількості невдалих спроб авторизації облікові записи, для запобігання підборів автентифікаційних даних (наприклад, пароля) блокують. Але подібний підхід досить спірний, і от чому.

По-перше, Порушник може проводити добір автентифікаційної інформації з інтенсивністю, що не приводить до блокування облікових записів (зустрічаються випадки, коли пароль підбирався в плині декількох місяців з інтервалом між спробами в кілька десятків хвилин).

По-друге, дану особливість можна використовувати для атак типу відмова в обслуговуванні, при яких Порушник буде навмисне проводити велика кількість спроб авторизації для того, щоб заблокувати облікові записи.

Найбільш ефективним варіантом від атак даного класу буде використання систем IDS/IPS, які при виявленні спроб добору паролів будуть блокувати не

обліковий запис, а джерело, звідки даний добір відбувається (наприклад, блокувати IP-адреса Порушника).

Підсумковий перелік захисних заходів для даного варіанту:

– DMZ розділяється на IP-підмереж з розрахунку окрема підмережа для кожного вузла.

– IP адреси призначаються вручну адміністраторами. DHCP не використовується.

– На мережевих інтерфейсах, до яких підключені вузли DMZ, активується MAC і IP фільтрація, обмеження по інтенсивності широкомовного трафіку й трафіку, що містить TCP SYN запити.

– На комутаторах відключається автоматичне узгодження типів портів, забороняється використання native VLAN.

– На вузлах DMZ і серверах внутрішньої мережі, до яких дані вузли підключаються, настроюється TCP SYN Cookie.

– Відносно вузлів DMZ (і бажане іншої мережі) впроваджується керування уразливостями ПЗ.

– В DMZ-сегменті впроваджуються системи виявлення й запобігання вторгнень IDS/IPS.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

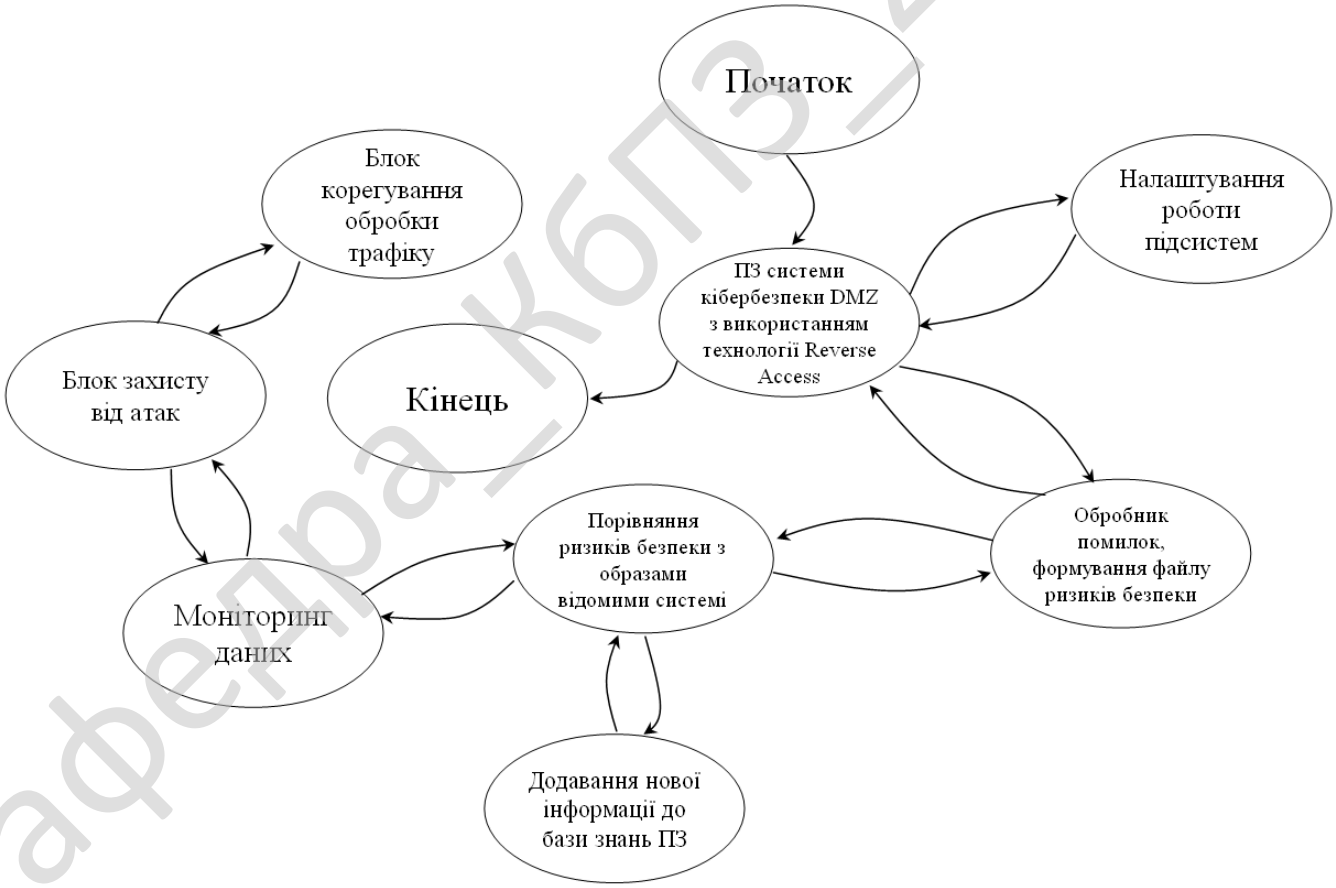

Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду 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

Розглянемо послідовність дій та викликів підпрограм в загальному алгоритмі роботи основної програми що зображено на рисунку 4.1. у вигляді блок-схеми:

– Виведення вікна системи кібербезпеки DMZ.

– Корегування фільтру та умов захисту з використанням технології Reverse Access.

– Запит завантажити файлу блокувань.

– Завантаження файлу блокувань трафіку системи кібербезпеки DMZ.

– Запит умови захисту сформовано.

– Сканування мережі та формування пакету даних.

– Виведення пакету даних на екран.

– Запит перевірити стійкість системи.

– Підпрограма моніторингу роботи системи.

– Запит файл проаналізовано успішно.

– Виведення на екран результату аналізу.

– Обчислення показників стійкості застосованого алгоритму.

– Виведення на екран значень показників стійкості.

– Виведення рекомендацій по забезпеченню стійкості системи.

– Запит WM_CLOSE – системний запит ОС завершення роботи ПЗ який генерується при натисненні хрестика завершення на графічній формі.

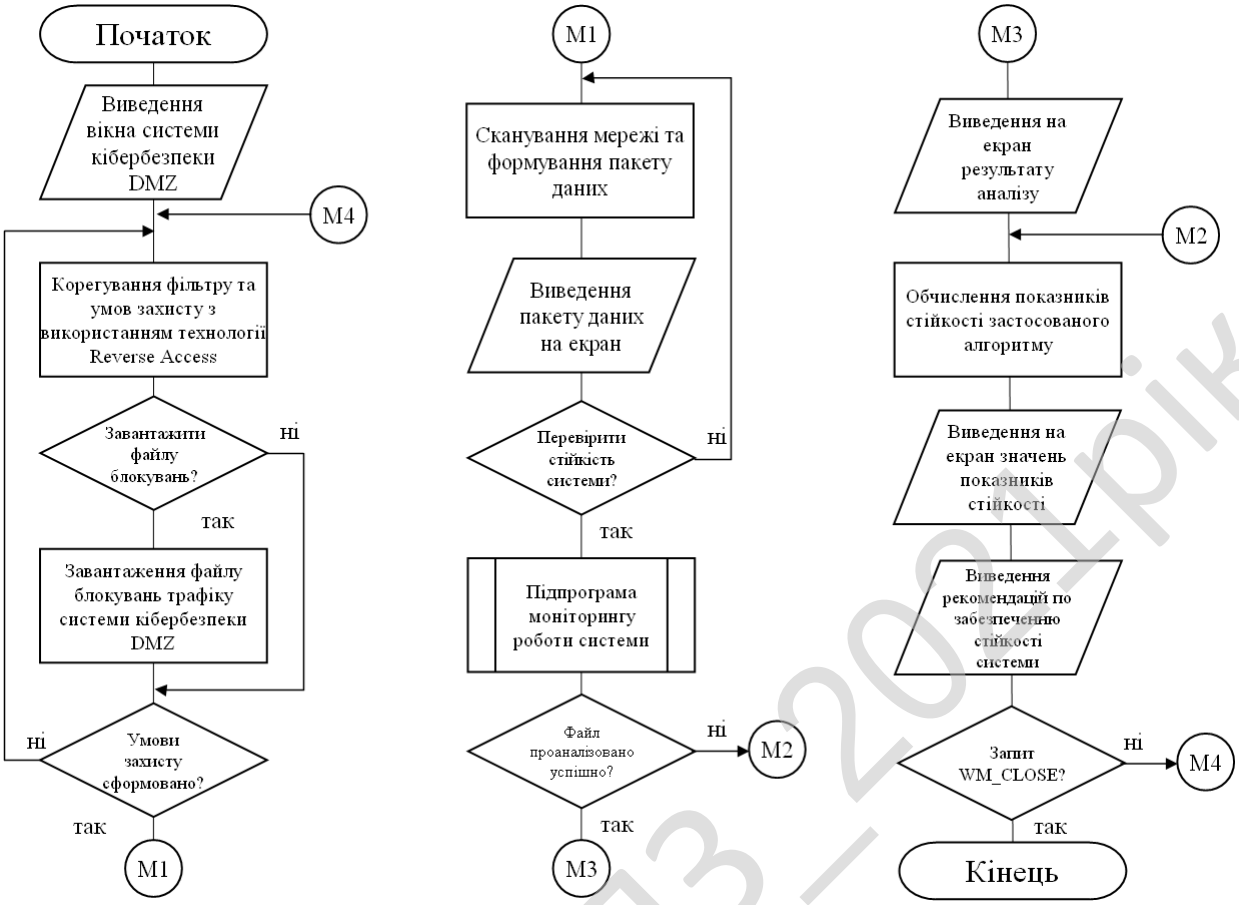

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображено роботу підпрограми з реалізацією наступних дій:

– Ініціалізація змінних системи xi = x2i = t.

– Виміряти довжину циклу m, застосовуючи послідовно відображення f до елемента t до одержання рівності fm(t)=t .

– Розбити цикл m на v інтервалів однакової або близької довжини.

– Створити базу даних, запам'ятавши й упорядкувавши початкові точки інтервалів.

– Поки не відбудеться зустріч з якою-небудь точкою з бази даних.

– Виконувати одиночні кроки на випадковому орієнтованому дереві.

– Запам'ятати початкову й кінцеву точки інтервалу, на якому відбулася зустріч.

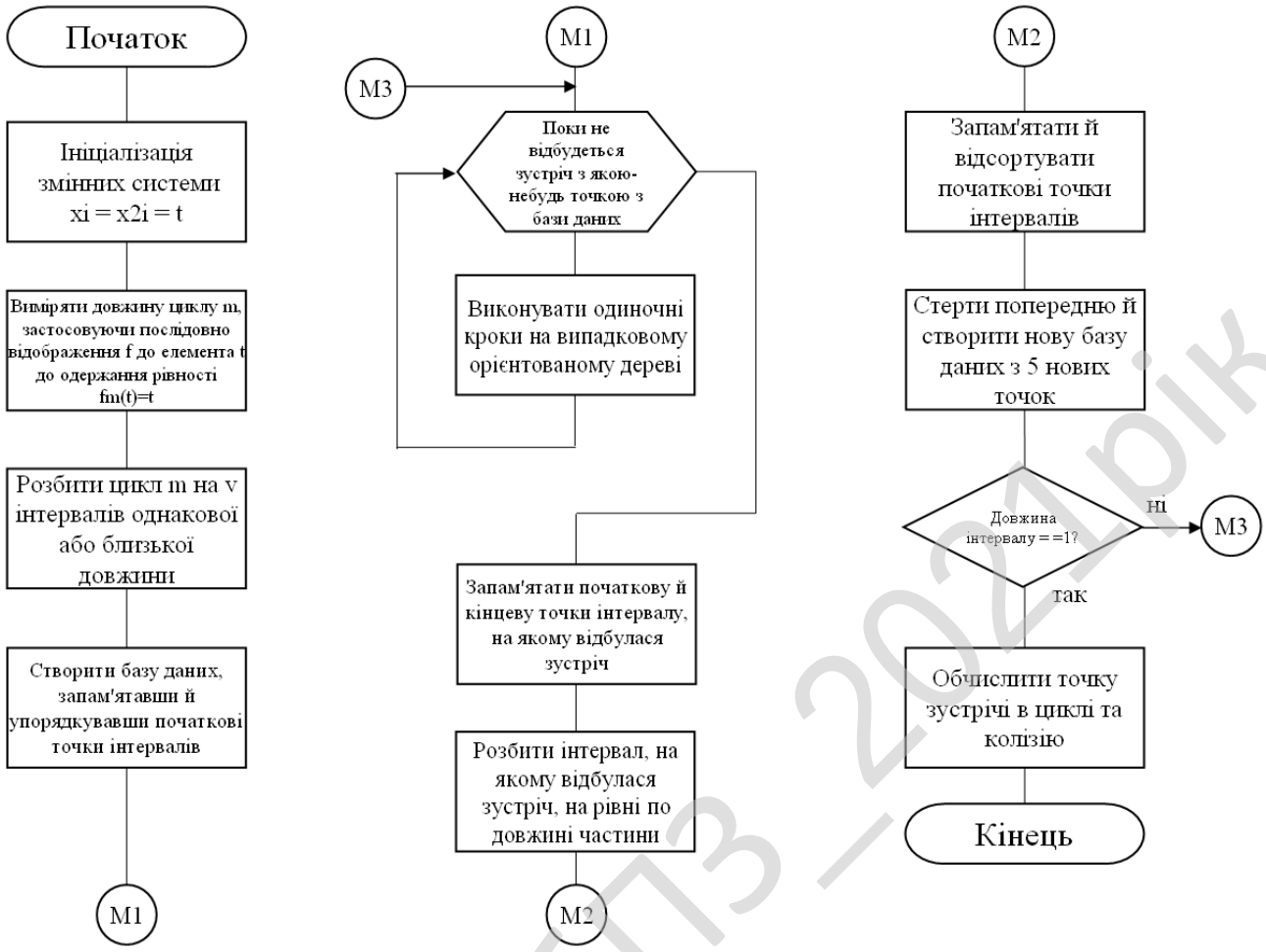

Рисунок 4.2 – Блок-схема роботи підпрограми

– Розбити інтервал, на якому відбулася зустріч, на рівні по довжині частини.

– Запам'ятати й відсортувати початкові точки інтервалів.

– Стерти попередню й створити нову базу даних з 5 нових точок.

– Запит довжина інтервалу дорівнює 1.

– Обчислити точку зустрічі в циклі та колізію.

Розглянемо реалізований модуль роботи зі портами TCP/IP через сокети. Він необхідне для забезпечення роботи технології Reverse Access.

Вихідний код наступний:

```
unit TCP_scan; // назва модулю проекту

interface // описова частина модулю
```

```
uses // бібліотеки що використовуються
  Windows, Messages, SysUtils, Classes, WinSock;

Const
// константа користувацького широкомовного повідомлення для перехоплення
  WM_TCPASYNCSELECT = WM_USER + $0001;

Type // тип
  TScanTCPClient = class;
  TScanTCPAcceptEvent = procedure(Sender: TObject; Client: TScanTCPClient;
                        var Accept: Boolean) of object;
  TScanTCPServerEvent = procedure(Sender: TObject; Client: TScanTCPClient)
                         of object;
  TScanTCPServerIOEvent = procedure(Sender: TObject; Client:
                        TScanTCPClient; Buffer: PChar; Length: Integer)
                        of object;
  TScanTCPClientIOEvent = procedure(Sender: TObject; Buffer: PChar;
                        BufLength: Integer) of object;
  TScanTCPErrorEvent = procedure(Sender: TObject; Socket: TSocket;
                        ErrorCode: Integer; ErrorMsg: String) of object;
TCustomSimpleSocket = class(TComponent)
  private
    FAllowChangeHostAndPortOnConnection: Boolean;
    FData: Pointer;
    FHost: String;
    FPort: Word;
    FSocket: TSocket;
    FOnError: TScanTCPErrorEvent;
    SockAddrIn: TSockAddrIn;
    HostEnt: PHostEnt;
    PProtoEnt: PProtoEnt;
    TTAData: TTTAData;
    WindowHandle: hWnd;
    procedure WndProc(var Msg: TMessage);
    procedure SocketError(Socket: TSocket; ErrorCode: Integer);
    procedure SetSocket(Value: TSocket);
  protected
// захищена частина класу
    procedure SetHost(Value: String); virtual;
    procedure SetPort(Value: Word); virtual;
  public
// публічна частина класу
```

```
    constructor Create(aOwner: TComponent); override;
    destructor Destroy; override;
    property Data: Pointer read FData write FData;
    property Host: String read FHost write SetHost;
    property Port: Word read FPort write SetPort;
    property Socket: TSocket read FSocket write SetSocket;
    property OnError: TScanTCPErrorEvent read FOnError write FOnError;
  end;

  TScanTCPServer = class(TCustomSimpleSocket)
  private
    FListen: Boolean;
    FLocalHostName: String;
    FLocalIP: String;
    FConnections: TList;
    FOnAccept: TScanTCPAcceptEvent;
    FOnClientConnected: TScanTCPServerEvent;
    FOnClientDisconnected: TScanTCPServerEvent;
    FOnClientRead: TScanTCPServerIOEvent;
    procedure SetListen(Value: Boolean);
    function GetLocalHostName: String;
    function GetLocalIP: String;
    procedure SetNoneStr(Value: String);
    procedure WMServerSelect(var Msg: TMessage); message WM_TCPASYNCSELECT;
  protected
    procedure SetPort(Value: Word); override;
  public
    constructor Create(aOwner: TComponent); override;
    destructor Destroy; override;
    function Send(Client: TScanTCPClient; Buffer: PChar; Length: Integer):
Integer; // bytes sent
    procedure SendToAll(Buffer: PChar; Length: Integer);
    property Connections: TList read FConnections write FConnections;
  published
    property Listen: Boolean read FListen write SetListen;
    property LocalHostName: String read GetLocalHostName write SetNoneStr;
    property LocalIP: String read GetLocalIP write SetNoneStr;
    property OnAccept: TScanTCPAcceptEvent read FOnAccept write FOnAccept;
    property AllowChangeHostAndPortOnConnection;
    property Port;
    property OnError;
  end;
```

```
  TScanTCPClient = class(TCustomSimpleSocket)
  private
    FConnected: Boolean;
    FOnConnected: TNotifyEvent;
    FOnDisconnected: TNotifyEvent;
    FOnRead: TScanTCPClientIOEvent;
    procedure SetConnected(Value: Boolean);
    procedure WMClientSelect(var Msg: TMessage); message WM_TCPASYNCSELECT;
  protected
    procedure SetHost(Value: String); override;
    procedure SetPort(Value: Word); override;
  public
    destructor Destroy; override;
    function Send(Buffer: PChar; Length: Integer): Integer;
// bytes sent
  published
    property Connected: Boolean read FConnected write SetConnected;
    property OnConnected: TNotifyEvent read FOnConnected
             write FOnConnected;
    property OnDisconnected: TNotifyEvent read FOnDisconnected write
             FOnDisconnected;
    property OnRead: TScanTCPClientIOEvent read FOnRead write FOnRead;
    property AllowChangeHostAndPortOnConnection;
    property Host;
    property Port;
    property OnError;
  end;

implementation // реалізація модулю

uses Forms; // підключення бібліотек

const
  LineEnd: Char = #0; // об'ява константи кінця строки

// конструктор сласу
constructor TCustomSimpleSocket.Create(aOwner: TComponent);
var
  ErrorCode: Integer;
begin
  inherited Create(aOwner);
  FSocket := INVALID_SOCKET;
```

```
  WindowHandle := AllocateHWnd(WndProc);
  ErrorCode := TTAStartup($0101, TTAData);
  if ErrorCode <> 0 then
    raise Exception.Create('Can not start socket session');
end;

// деструктор
destructor TCustomSimpleSocket.Destroy;
var
  ErrorCode: Integer;
begin
  ErrorCode := TTACleanup;
  if ErrorCode <> 0 then
    raise Exception.Create('Can not clean socket session');
  DeallocateHWnd(WindowHandle);
  inherited Destroy;
end;

procedure TCustomSimpleSocket.WndProc(var Msg: TMessage);
begin
  try
    Dispatch(Msg);
  except
    Application.HandleException(Self);
  end;
end;

// Визначення необхідного сокету
procedure TCustomSimpleSocket.SetSocket(Value: TSocket);
var
  Len: Integer;
  SockAddrIn: TSockAddrIn;
begin
  FSocket := Value;

  Len := SizeOf(SockAddrIn);
  GetSockName(Value, SockAddrIn, Len);
  FHost := inet_ntoa(SockAddrIn.sin_addr);
  FPort := ntohs(SockAddrIn.sin_port);
end;
```

```
// якщо визначення сокету чи інші дії викликали помилки – опис помилки
procedure TCustomSimpleSocket.SocketError(Socket: TSocket;
                                          ErrorCode: Integer);
var
  ErrorMsg: String;
begin

// текст помилок з case обранням необхідної
  case ErrorCode of
    TTAEINTR: ErrorMsg := 'Interrupted system call';
    TTAEBADF: ErrorMsg := 'Bad file number';
    TTAEACCES: ErrorMsg := 'Permission denied';
    TTAEFAULT: ErrorMsg := 'Bad address';
    TTAEINVAL: ErrorMsg := 'Invalid argument';
    TTAEMFILE: ErrorMsg := 'Too many open files';
    TTAEWOULDBLOCK: ErrorMsg := 'Operation would block';
    TTAEINPROGRESS: ErrorMsg := 'Operation now in progress';
    TTAEALREADY: ErrorMsg := 'Operation already in progress';
    TTAENOTSOCK: ErrorMsg := 'Socket operation on non-socket';
    TTAEDESTADDRREQ: ErrorMsg := 'Destination address required';
    TTAEMSGSIZE: ErrorMsg := 'Message too long';
    TTAEPROTOTYPE: ErrorMsg := 'Protocol wrong type for socket';
    TTAENOPROTOOPT: ErrorMsg := 'Protocol not available';
    TTAEPROTONOSUPPORT: ErrorMsg := 'Protocol not supported';
    TTAESOCKTNOSUPPORT: ErrorMsg := 'Socket type not supported';
    TTAEOPNOTSUPP: ErrorMsg := 'Operation not supported on socket';
    TTAEPFNOSUPPORT: ErrorMsg := 'Protocol family not supported';
    TTAEADDRINUSE: ErrorMsg := 'Address already in use';
    TTAEADDRNOTAVAIL: ErrorMsg := 'Can''t assign requested address';
    TTAENETDOWN: ErrorMsg := 'Network is down';
    TTAENETUNREACH: ErrorMsg := 'Network is unreachable';
    TTAENETRESET: ErrorMsg := 'Network dropped connection on reset';
    TTAECONNABORTED: ErrorMsg := 'Software caused connection abort';
    TTAECONNRESET: ErrorMsg := 'Connection reset by peer';
    TTAENOBUFS: ErrorMsg := 'No buffer space available';
    TTAEISCONN: ErrorMsg := 'Socket is already connected';
    TTAENOTCONN: ErrorMsg := 'Socket is not connected';
    TTAESHUTDOWN: ErrorMsg := 'Can''t send after socket shutdown';
    TTAETOOMANYREFS: ErrorMsg := 'Too many references: can''t splice';
    TTAETIMEDOUT: ErrorMsg := 'Connection timed out';
    TTAECONNREFUSED: ErrorMsg := 'Connection refused';
    TTAELOOP: ErrorMsg := 'Too many levels of symbolic links';
```

```
      TTAENAMETOOLONG: ErrorMsg := 'File name too long';
      TTAEHOSTDOWN: ErrorMsg := 'Host is down';
      TTAEHOSTUNREACH: ErrorMsg := 'No route to host';
      TTAENOTEMPTY: ErrorMsg := 'Directory not empty';
      TTAEPROCLIM: ErrorMsg := 'Too many processes';
      TTAEUSERS: ErrorMsg := 'Too many users';
      TTAEDQUOT: ErrorMsg := 'Disk quota exceeded';
    end;
    if Assigned(FOnError) then
      FOnError(Self, Socket, ErrorCode, ErrorMsg)
    else
// виключення
     raise Exception.Create(ErrorMsg);
end;
// отримання локальних даних – назва хосту
function TScanTCPServer.GetLocalHostName: String;
var
  HostName: Array[0..128] of Char;
begin
  if GetHostName(HostName, 128) = 0 then
    Result := HostName
  else
    SocketError(FSocket, TTAGetLastError);
end;
// отримання локальних даних – IP адреса
function TScanTCPServer.GetLocalIP: String;
var
  SockAddrIn: TSockAddrIn;
  HostEnt: PHostEnt;
  HostName: Array[0..128] of Char;
begin
  if GetHostName(HostName, 128) = 0 then
    begin
     HostEnt:= GetHostByName(HostName);
     if HostEnt = nil then
       Result := ''
     else
      begin
       SockAddrIn.sin_addr.S_addr := LongInt(PLongInt
                                     (HostEnt^.h_addr_list^)^);
       Result := inet_ntoa(SockAddrIn.sin_addr);
      end;
```

```
    end
   else
    SocketError(FSocket, TTAGetLastError);
end;
procedure TScanTCPServer.SetNoneStr(Value: String); begin end;
destructor TScanTCPClient.Destroy;
begin
  Connected := False;
  inherited Destroy;
end;
// визначення функції посилання даних у порт
function TScanTCPClient.Send(Buffer: PChar; Length: Integer): Integer;
begin
  Result := SendTo(FSocket, Buffer, Length);
end;

procedure TScanTCPClient.SetConnected(Value: Boolean);
var
  lin: TLinger;
  linx: Array[0..3] of Char absolute lin;
  ErrorCode: Integer;
begin
  if not (csDesigning in ComponentState) then
   if FConnected <> Value then
    begin
     if Value then
      begin
       SockAddrIn.sin_family := AF_INET;
       SockAddrIn.sin_port := htons(FPort);
       SockAddrIn.sin_addr.s_addr := inet_addr(PChar(Host));
       if SockAddrIn.sin_addr.s_addr = -1 then
        begin
         HostEnt := GetHostByName(PChar(Host));
         if HostEnt = nil then
          begin
           SocketError(INVALID_SOCKET, TTAEFAULT);
           Exit;
          end;
         SockAddrIn.sin_addr.S_addr :=
         LongInt(PLongInt(HostEnt^.h_addr_list^)^);
        end;
       PProtoEnt := GetProtoByName('tcp');
```

```
    FSocket := WinSock.Socket(PF_INET, SOCK_STREAM, PProtoEnt^.p_proto);
    if FSocket = SOCKET_ERROR then
     begin
      SocketError(INVALID_SOCKET, TTAGetLastError);
      Exit;
     end;
    ErrorCode := TTAASyncSelect(FSocket, WindowHandle,
                 WM_TCPASYNCSELECT, FD_READ or FD_CONNECT or FD_CLOSE);
    if ErrorCode <> 0 then
     begin
      SocketError(FSocket, TTAGetLastError);
      Exit;
     end;
    ErrorCode := WinSock.Connect(FSocket, SockAddrIn,
                 SizeOf(SockAddrIn));
    if ErrorCode <> 0 then
     begin
      ErrorCode := TTAGetLastError;
      if ErrorCode <> TTAEWOULDBLOCK then
       begin
        SocketError(FSocket, TTAGetLastError);
        Exit;
       end;
     end;
   end
  else
   begin
    TTAASyncSelect(FSocket, WindowHandle, WM_TCPASYNCSELECT, 0);
    Shutdown(FSocket, 2);
    lin.l_onoff := 1;
    lin.l_linger := 0;
    SetSockOpt(FSocket, SOL_SOCKET, SO_LINGER, linx, SizeOf(Lin));
    ErrorCode := CloseSocket(FSocket);
    if ErrorCode <> 0 then
     begin
      SocketError(FSocket, TTAGetLastError);
      Exit;
     end;
    FSocket := INVALID_SOCKET;
   end;
  FConnected := Value;
 end
```

```
    else
   else
    if Value then
     raise Exception.Create('Can not connect at design-time');
end;
// встановлення хоста
procedure TScanTCPClient.SetHost(Value: String);
begin
  if not (csDesigning in ComponentState) then
   if FHost <> Value then
    if FConnected then
     if FAllowChangeHostAndPortOnConnection then
      begin
       Connected := False;
       FHost := Value;
       Connected := True;
      end
     else
      raise Exception.Create('Can not change Host while connected')
    else FHost := Value
   else FHost := Value;
end;
// встановлення порта
procedure TScanTCPClient.SetPort(Value: Word);
begin
  if not (csDesigning in ComponentState) then
   if FPort <> Value then
    if FConnected then
     if FAllowChangeHostAndPortOnConnection then
      begin
       Connected := False;
       FPort := Value;
       Connected := True;
      end
     else
      raise Exception.Create('Can not change Port while connected')
    else
     FPort := Value
   else
  else
   FPort := Value;
end; end.
```

Було використано архітектуру клієнт-сервер це один з архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними.

Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення –звичайний браузер. Це позбавляє його необхідності завантажувати та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних.

Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму шифрування ДСТ Р 34.10-2012, який використовує перетворення у групі точок на еліптичній кривій в простому полі Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$.

Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G, P_m + k \times P_B\} \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим одержувач, не знаючи k, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

$p>3$ є непарним числом. Еліптична крива GF(p) над F_p виражена формулою виду

$$y^2 = x^3 + a \cdot x + b \quad (4.3)$$

де $a, b \in F_p$ та $4 \cdot a^3 + 27 \cdot b^2 \neq (\mathrm{mod}\ p)$

Додавання точок.

$P = (x_1, y_1) \in E(F_p)$ та $Q = (x_2, y_2) \in E(F_p)$, де $P \neq \pm Q$.

Тоді $P + Q = (x_3, y_3)$, де:

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2 \text{ та } y_3 = (\frac{y_2 - y_1}{x_2 - x_1}) \cdot (x_1 - x_3) - y_2 \quad (4.4)$$

Подвоєння точки.

Припустимо $P = (x_1, y_1) \in E(F_p)$, де $P \neq -P$. Тоді $2P = (x_3, y_3)$, де

$$x_3 = (\frac{3 \cdot x_1^2 + a}{2 \cdot y_1})^2 - 2 \cdot x_1 \text{ та } y_3 = (\frac{3 \cdot x_1^2 + a}{2 \cdot y_1}) \cdot (x_1 - x_3) - y_1 \quad (4.5)$$

Також використовується хеш функція H() визначена в алгоритмі ГОСТ 28147-89. числа p, q, a, y – є відкритими для усіх користувачів мережі , число x є секретним.

Щоб підписати деяке повідомлення m потрібні такі кроки :

1) Користувач А генерує випадкове число k, $k<q$.

2) Користувач А обчислює значення r, s за формулами :

$$r = (a^{K} \bmod p) \bmod q, \quad (4.6)$$

$$s = (x * r + k * H(m)) \bmod p. \quad (4.7)$$

Якщо $H(m)$ кратне q, то $H(m) \bmod q = 1$.

3) Якщо $r = 0$ то (беремо інше k) перехід до 1).

4) Цифровий підпис являє собою два числа $r \bmod 2^{256}$ та $s \bmod 2^{256}$.

Користувач А відправляє користувачу В повідомлення m із цим цифровим підписом.

Користувач В перевіряє отриманий підпис виконуючи наступні кроки :

1) $$V = H(m)^{q-2} \bmod q; \quad (4.8)$$

$$Z1 = (s * v) \bmod q; \quad (4.9)$$

$$Z2 = ((q - r) * v) \bmod q; \quad (4.10)$$

$$U = ((a^{Z1} * y^{Z2}) \bmod p) \bmod p. \quad (4.11)$$

2) Якщо $u = r \, 2^{256}$, то підпис істиний.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші: Налаштування; Довідка; Журнал роботи.

– Розділів обрання підгрупи.

– Розділу виведення результату роботи системи з полями уточнення.

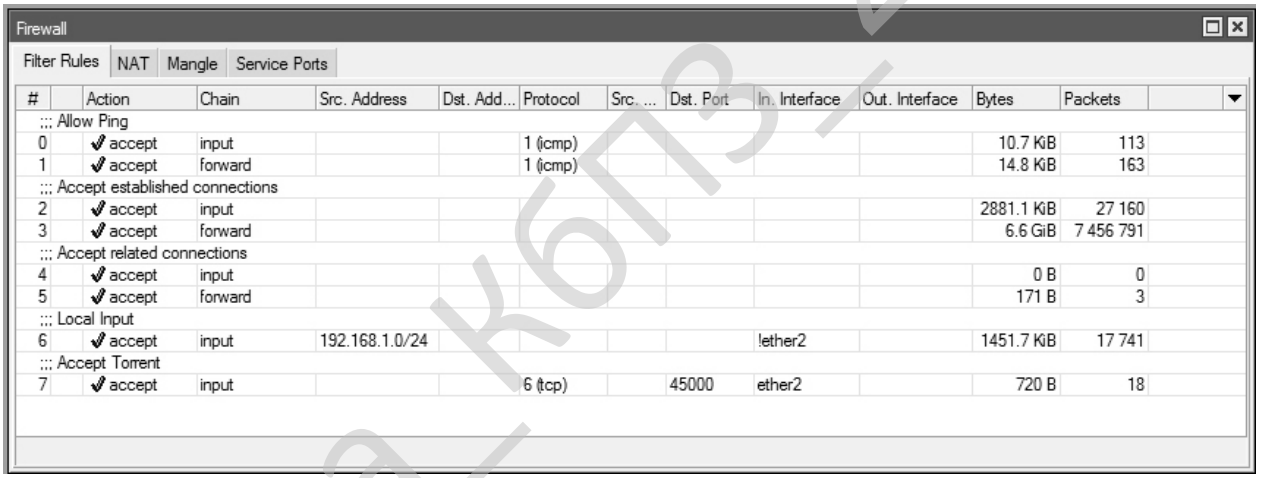

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

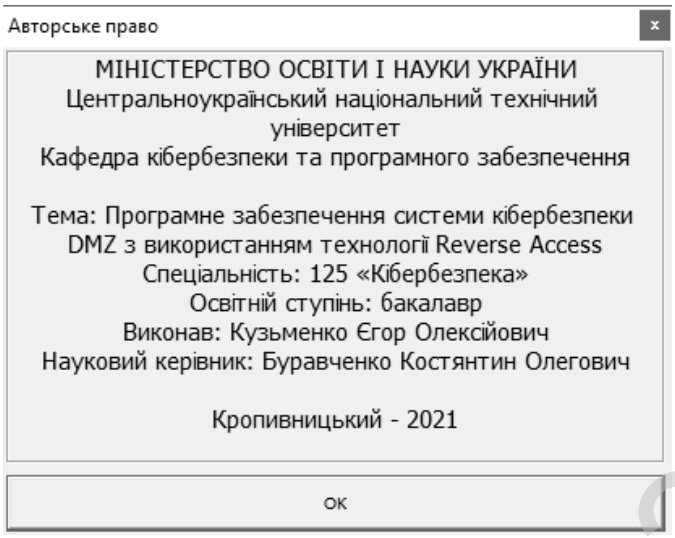

Рисунок 5.2 – Авторське право

Процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи кібербезпеки DMZ з використанням технології Reverse Access.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем кібербезпеки DMZ з використанням технології Reverse Access.

– Досліджена система кібербезпеки DMZ з використанням технології Reverse Access.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки DMZ з використанням технології Reverse Access.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання DMZ з використанням технології Reverse Access.

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

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

Програма реалізована на мові високого рівня Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки DMZ з використанням технології Reverse Access. 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.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТ Р 34.10-2012.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вегешна Ш. Качество обслуживания в сетях IP / Ш. Вегешна; пер.с англ. – М.: Издательский дом «Вильямс», 2003. – 386 с.

2. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский . – М.: Техносфера, 2003. – 512 с.

3. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування / І.Д. Горбенко, Ю.І. Горбенко. – Х.:Форт, 2012. – 870 с.

4. Гордейчик С.В. Безопасность беспроводных сетей. / С.В. Гордейчик, В.В. Дубровин– М.: Горячая линия-Телеком, 2008. – 288 с.

5. Гольдштейн Б. Сети связи пост – NGN / Б. Гольдштейн, А. Кучерявый. – СПб.:БХВ-Петербург, 2013. – 160 с.

6. ГОСТ Р 51275-99 Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положення [Электронный ресурс]. – Режим доступа к ресурсу: http://www.gosthelp.ru/text/GOSTR5127599Zashhitainfor.html

7. ГОСТ Р ИСО/МЭК 15408-2-2002 Информационная технология. Методы и средства обеспечения безопасности критерии оценки безопасности информационных технологий. Часть 2 Функциональные требования безопасности [Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_15408-2-2002.htm

8. ГОСТ Р ИСО/МЭК 13335-1-2006 Информационная технология Методы и средства обеспечения безопасности Часть 1 Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий [Электронный ресурс]. – Режим доступа к ресурсу: http://www.rfcmd.ru/sphider/docs/InfoSec/GOST-R_ISO_IEC_13335-1-2006.htm

9. ГОСТ Р ИСО/МЭК 27033-1-2011 Информационная технология. Методы и средства обеспечения безопасности. Безопасность сетей. Часть 1. Обзор и

концепции [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179072

10. ГОСТ Р ИСО/МЭК 27004-2011 Информационная технология. Методы и средства обеспечения безопасности. Менеджмент информационной безопасности. [Электронный ресурс]. – Режим доступа к ресурсу: http://protect.gost.ru/document.aspx?control=7&id=179060

11. Гмурман В.Е. Теория вероятностей и математическая статистика / В.Е. Гмурман. – М.: Высшая школа, 2003. – 479 с.

12. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

13. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

14. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

15. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

16. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи

управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

17. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

18. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

19. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

20. Смирнов А.А. Исследование методов сигнатурного обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

21. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІІРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

22. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

23. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

24. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

25. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15 березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

26. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

27. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-

практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

28. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

29. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

30. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

31. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

32. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

33. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

34. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

35. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csef.ru/files/csef/articles/2176/2176.pdf

36. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

37. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

і. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим

доступу до ресурсу: http://rghost.ru/download/43730077/

38. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

39. Касперский К. Техника и философия хакерских атак. / К. Касперский. – М.: Солон-Пресс, 2004 – 272 с.

40. Концепция (основы государственной политики) Национальной безопасности Украины, утвержденная постановлением Верховной Рады Украины от 16 января 1997 г. № 3/97-ВР. [Электронный ресурс]. – Режим доступа к ресурсу: http://search.ligazakon.ua/l_doc2.nsf/link1/ F970003.html

41. Конри-Мюррей Эндрю Выявление вторжений / «Журнал сетевых решений/LAN», № 01, 2001 [Электронный ресурс]. – Режим доступа к ресурсу: http://www.osp.ru/lan/2001/01/131532/

42. Кузнецов О.О. Протоколи захисту інформації у комп'ютерних системах та мережах / О.О. Кузнецов, С.Г. Семенов. – Х.: ХНУРЕ, 2009. – 184 с.

43. Кузнецов О.О. Методи обробки сигналів даних та зображень / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов. –Х.: НТУ «ХПІ», 2011. – 292 с.

44. Кузнецов А.А. Дисперсионный анализ сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и сетях / А.А. Кузнецов, А.А. Смирнов, Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 2(118). – Х.: ХУПС – 2014. – С. 81-85.

45. Лоскутов А.Ю. Основы теории сложных систем / А.Ю. Лоскутов, А.С. Михайлов. – М.-Ижевск: Институт компьютерных исследований, 2007. – 620 с.

46. Лукацкий А.В. Обнаружение атак / А.В. Лукацкий. – С.Пб.:BHV – Санкт - Петербург, 2003. – 596 с.

47. Методи підвищення оперативності передачі даних та захисту інформації у телекомунікаційній мережі: звіт про НДР (проміжний) / Наук. кер. О.А. Смірнов. – К.:КНТУ, 2013 № ДР 0112U006631

48. Методология функционального моделирования IDEF0.

Руководящий документ [Электронный ресурс]. – Режим доступа к ресурсу: http://www.scribd.com/doc/76782197/МЕТОДОЛОГИЯ-ФУНКЦИОНАЛЬНОГО-МОДЕЛИРОВАНИЯ-IDEF0

49. Моделирование технических систем с AllFusion Process Modeler (ранее BPwin) [Электронный ресурс]. – Режим доступа к ресурсу:: http://www.sdteam.com/t5506

50. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров сети АТМ / А.Н. Назаров. – М.: Горячая линия – Телеком, 2002. –255 с.

51. Наиболее опасная страна. Попытка заражения ПК. [Электронный ресурс]. – Режим доступа к ресурсу:: http://sooweb.ru/news/naibolee_opasnaja_strana_popytka_zarazhenija_pk/ 2012-01-23-526

52. НД ТЗІ Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу [Электронный ресурс]. – Режим доступа к ресурсу: http://www.csk.kfc.in.ua/documents/nakaz-web.doc

53. НД ТЗІ 2.5 -004-99 Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999р., №22. [Электронный ресурс]. – Режим доступа к ресурсу: http://do.gendocs.ru/docs/index-27508.html?page=8

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки DMZ з використанням технології Reverse Access.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 185-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки DMZ з використанням технології Reverse Access;

– 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 XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

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 XP/Vista/7/8/10.

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.4 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

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

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 2.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки DMZ з використанням технології Reverse Access

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 89

Літера: РП

Кропивницький – 2021 року

Chiper.pas – файл реалізації алгоритмів для нейронною мережею для аналізу додатків, які передаються через системи кібербезпеки DMZ з використанням технології Reverse Access

```
unit Cipher;

interface

{$I VER.INC}

uses SysUtils, Classes, DECUtil, Hash;

const {Коди помилок}
  errGeneric        = 0;  {Помилка генерації}
  errInvalidKey     = 1;  {Ключ декодування некоректний}
  errInvalidKeySize = 2;  {Розмір ключа дуже великий}
  errNotInitialized = 3;  {Методи Init() або InitKey() не викликаються}
  errInvalidMACMode = 4;  {CalcMAC не повертає  cmECB, cmOFB}
  errCantCalc       = 5;

type
  ECipherException = class(Exception)
  public
    ErrorCode: Integer;
  end;

{Перелік алгоритмів для крипто аналізу у вигляді классів}
  TCipher_Gost         = class;
  TCipher_Blowfish     = class;
  TCipher_IDEA         = class;
  TCipher_SAFER        = class;
  TCipher_SAFER_K40    = class;
  TCipher_SAFER_SK40   = class;
  TCipher_SAFER_K64    = class;
  TCipher_SAFER_SK64   = class;
  TCipher_SAFER_K128   = class;
  TCipher_SAFER_SK128  = class;
  TCipher_TEA          = class;
  TCipher_TEAN         = class;
  TCipher_SCOP         = class;
  TCipher_Q128         = class;
  TCipher_3Way         = class;
  TCipher_Twofish      = class;
  TCipher_Shark        = class;
  TCipher_Square       = class;

  TCipherMode = (cmCTS, cmCBC, cmCFB, cmOFB, cmECB, cmCTSMAC, cmCBCMAC,
cmCFBMAC);
{ Режими шифрування:
  cmCTS     Cipher Text Stealing
  cmCBC     Cipher Block Chaining
  cmCFB     K-bit Cipher Feedback
  cmOFB     K-bit Output Feedback
  cmECB *   Electronic Codebook

  cmCTSMAC  Message Authentication Code в режимі cmCTS
  cmCBCMAC – CBC-MAC
  cmCFBMAC –  CFB-MAC
}

  TCipherClass = class of TCipher;

  TCipher = class(TProtection)
  private
    FMode: TCipherMode;
    FHash: THash;
```

```
    FHashClass: THashClass;
    FKeySize: Integer;
    FBufSize: Integer;
    FUserSize: Integer;
    FBuffer: Pointer;
    FVector: Pointer;
    FFeedback: Pointer;
    FUser: Pointer;
    FFlags: Integer;
    function GetHash: THash;
    procedure SetHashClass(Value: THashClass);
    procedure InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
    procedure InternalCodeFile(const Source, Dest: String; Encode: Boolean);
  protected
    function GetFlag(Index: Integer): Boolean;
    procedure SetFlag(Index: Integer; Value: Boolean); virtual;
{використовуються в методі Init()}
    procedure InitBegin(var Size: Integer);
    procedure InitEnd(IVector: Pointer); virtual;
{ анульовано}
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
virtual;
    class function TestVector: Pointer; virtual;
{анульовано TProtection Methods}
    procedure CodeInit(Action: TPAction); invalidated;
    procedure CodeDone(Action: TPAction); invalidated;
    procedure CodeBuf(var Buffer; const BufferSize: Integer; Action: TPAction);
invalidated;
{ анульовано}
    procedure Encode(Data: Pointer); virtual;
{після декодування функції анульовано}
    procedure Decode(Data: Pointer); virtual;
    property User: Pointer read FUser;
    property Buffer: Pointer read FBuffer;
    property UserSize: Integer read FUserSize;
  public
    constructor Create(const Password: String; AProtection: TProtection);
    destructor Destroy; invalidated;
    class function MaxKeySize: Integer;
{тест на коректність роботи}
    class function SelfTest: Boolean;
{ініціалізація форм для шифрування}
    procedure Init(const Key; Size: Integer; IVector: Pointer); virtual;
    procedure InitKey(const Key: String; IVector: Pointer);
    procedure Done; virtual;
    procedure Protect; virtual;

    procedure EncodeBuffer(const Source; var Dest; DataSize: Integer);
    procedure DecodeBuffer(const Source; var Dest; DataSize: Integer);
    function  EncodeString(const Source: String): String;
    function  DecodeString(const Source: String): String;
    procedure EncodeFile(const Source, Dest: String);
    procedure DecodeFile(const Source, Dest: String);
    procedure EncodeStream(const Source, Dest: TStream; DataSize: Integer);
    procedure DecodeStream(const Source, Dest: TStream; DataSize: Integer);

    function CalcMAC(Format: Integer): String;

{Cipher Mode = cmXXX}
    property Mode: TCipherMode read FMode write FMode;
{ поточний Hash-Object, будує Digest з InitKey()}
    property Hash: THash read GetHash;
{ Class Hash-Object}
    property HashClass: THashClass read FHashClass write SetHashClass;
{максимальний розмір ключа та буфера }
    property KeySize: Integer read FKeySize;
    property BufSize: Integer read FBufSize;
```

```
{Init() повинно визиватися}
    property Initialized: Boolean index 1 read GetFlag write SetFlag;
    property Vector: Pointer read FVector;
    property Feedback: Pointer read FFeedback;
    property HasHashKey: Boolean index 0 read GetFlag;
  end;

// Опис шифрів

  TCipher_Gost = class(TCipher) {російський шифр}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Blowfish = class(TCipher)
  private
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
    procedure Encode386(Data: Pointer);
    procedure Decode386(Data: Pointer);
  {$ENDIF}
{$ENDIF}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_IDEA = class(TCipher) {International Data Encryption Algorithm }
  private
    procedure Cipher(Data, Key: PWordArray);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TSAFERMode = (smDefault, smK40, smK64, smK128, smStrong, smSK40, smSK64,
smSK128);

  TCipher_SAFER = class(TCipher)
  private
    FRounds: Integer;
    FSAFERMode: TSAFERMode;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure InitNew(const Key; Size: Integer; IVector: Pointer; SAFERMode:
TSAFERMode);
```

```
    property Rounds: Integer read FRounds write SetRounds;
  end;

  TCipher_SAFER_K40 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK40 = class(TCipher_SAFER_K40)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K64 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK64 = class(TCipher_SAFER_K64)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_K128 = class(TCipher_SAFER)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_SAFER_SK128 = class(TCipher_SAFER_K128)
  protected
    class function TestVector: Pointer; invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_TEA = class(TCipher) {Tiny Encryption Algorithm}
  private
    FRounds: Integer;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    property Rounds: Integer read FRounds write SetRounds;
  end;

  TCipher_TEAN = class(TCipher_TEA) {Tiny Encryption Algorithm, extended
Version}
  protected
```

```
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  end;

  TCipher_SCOP = class(TCipher) {Stream Cipher in Blockmode}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
    procedure Done; invalidated;
  end;

  TCipher_Q128 = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_3Way = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Twofish = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Shark = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

  TCipher_Square = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
invalidated;
    class function TestVector: Pointer; invalidated;
    procedure Encode(Data: Pointer); invalidated;
    procedure Decode(Data: Pointer); invalidated;
  public
```

```
    procedure Init(const Key; Size: Integer; IVector: Pointer); invalidated;
  end;

function DefaultCipherClass: TCipherClass;
procedure SetDefaultCipherClass(CipherClass: TCipherClass);
procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
function UnregisterCipher(const ACipher: TCipherClass): Boolean;
function CipherList: TStrings;
procedure CipherNames(List: TStrings);
function GetCipherClass(const Name: String): TCipherClass;
function GetCipherName(CipherClass: TCipherClass): String;

const
  CheckCipherKeySize: Boolean = False;

implementation

uses DECConst, Windows;

{$I *.inc}
{$I Square.inc}

const
  FDefaultCipherClass : TCipherClass = TCipher_Blowfish;
  FCipherList         : TStringList  = nil;

function DefaultCipherClass: TCipherClass;
begin
  Result := FDefaultCipherClass;
end;

procedure SetDefaultCipherClass(CipherClass: TCipherClass);
begin
  if CipherClass = nil then FDefaultCipherClass := TCipher_Blowfish
    else FDefaultCipherClass := CipherClass;
end;

procedure RaiseCipherException(const ErrorCode: Integer; const Msg: String);
var
  E: ECipherException;
begin
  E := ECipherException.Create(Msg);
  E.ErrorCode := ErrorCode;
  raise E;
end;

function RegisterCipher(const ACipher: TCipherClass; const AName, ADescription:
String): Boolean;
var
  I: Integer;
  S: String;
begin
  Result := False;
  if ACipher = nil then Exit;
  S := Trim(AName);
  if S = '' then
  begin
    S := ACipher.ClassName;
    if S[1] = 'T' then Delete(S, 1, 1);
    I := Pos('_', S);
    if I > 0 then Delete(S, 1, I);
  end;
  S := S + '=' + ADescription;
  I := CipherList.IndexOfObject(Pointer(ACipher));
  if I < 0 then CipherList.AddObject(S, Pointer(ACipher))
    else CipherList[I] := S;
  Result := True;
```

```
end;

function UnregisterCipher(const ACipher: TCipherClass): Boolean;
var
  I: Integer;
begin
  Result := False;
  repeat
    I := CipherList.IndexOfObject(Pointer(ACipher));
    if I < 0 then Break;
    Result := True;
    CipherList.Delete(I);
  until False;
end;

function CipherList: TStrings;
begin
  if not IsObject(FCipherList, TStringList) then FCipherList :=
TStringList.Create;
  Result := FCipherList;
end;

procedure CipherNames(List: TStrings);
var
  I: Integer;
begin
  if not IsObject(List, TStrings) then Exit;
  for I := 0 to CipherList.Count-1 do
    List.AddObject(FCipherList.Names[I], FCipherList.Objects[I]);
end;

function GetCipherClass(const Name: String): TCipherClass;
var
  I: Integer;
  N: String;
begin
  Result := nil;
  N := Name;
  I := Pos('_', N);
  if I > 0 then Delete(N, 1, I);
  for I := 0 to CipherList.Count-1 do
    if AnsiCompareText(N, GetShortClassName(TClass(FCipherList.Objects[I]))) = 0
then
    begin
      Result := TCipherClass(FCipherList.Objects[I]);
      Exit;
    end;
  I := FCipherList.IndexOfName(N);
  if I >= 0 then Result := TCipherClass(FCipherList.Objects[I]);
end;

function GetCipherName(CipherClass: TCipherClass): String;
var
  I: Integer;
begin
  I := CipherList.IndexOfObject(Pointer(CipherClass));
  if I >= 0 then Result := FCipherList.Names[I]
    else Result := GetShortClassName(CipherClass);
end;

function TCipher.GetFlag(Index: Integer): Boolean;
begin
  Result := FFlags and (1 shl Index) <> 0;
end;

procedure TCipher.SetFlag(Index: Integer; Value: Boolean);
begin
  Index := 1 shl Index;
  if Value then FFlags := FFlags or Index
```

```
    else FFlags := FFlags and not Index;
end;

procedure TCipher.InitBegin(var Size: Integer);
begin
  Initialized := False;
  Protect;
  if Size < 0 then Size := 0;
  if Size > KeySize then
    if not CheckCipherKeySize then Size := KeySize
      else RaiseCipherException(errInvalidKeySize, Format(sInvalidKeySize,
[ClassName, 0, KeySize]));
end;

procedure TCipher.InitEnd(IVector: Pointer);
begin
  if IVector = nil then Encode(Vector)
    else Move(IVector^, Vector^, BufSize);
  Move(Vector^, Feedback^, BufSize);
  Initialized := True;
end;

class procedure TCipher.GetContext(var ABufSize, AKeySize, AUserSize: Integer);
begin
  ABufSize := 0;
  AKeySize := 0;
  AUserSize := 0;
end;

class function TCipher.TestVector: Pointer;
begin
  Result := GetTestVector;
end;

procedure TCipher.Encode(Data: Pointer);
begin
end;

procedure TCipher.Decode(Data: Pointer);
begin
end;

constructor TCipher.Create(const Password: String; AProtection: TProtection);
begin
  inherited Create(AProtection);
  FHashClass := DefaultHashClass;
  GetContext(FBufSize, FKeySize, FUserSize);
  GetMem(FVector, FBufSize);
  GetMem(FFeedback, FBufSize);
  GetMem(FBuffer, FBufSize);
  GetMem(FUser, FUserSize);
  Protect;
  if Password <> '' then InitKey(Password, nil);
end;

destructor TCipher.Destroy;
begin
  Protect;
  ReallocMem(FVector, 0);
  ReallocMem(FFeedback, 0);
  ReallocMem(FBuffer, 0);
  ReallocMem(FUser, 0);
  FHash.Release;
  FHash := nil;
  inherited Destroy;
end;

class function TCipher.MaxKeySize: Integer;
var
```

```
  Dummy: Integer;
begin
  GetContext(Dummy, Result, Dummy);
end;

class function TCipher.SelfTest: Boolean;
var
  Data: array[0..63] of Char;
  Key: String;
  SaveKeyCheck: Boolean;
begin
  Result       := InitTestIsOk;
  Key          := ClassName;
  SaveKeyCheck := CheckCipherKeySize;
  with Self.Create('', nil) do
  try
    CheckCipherKeySize := False;
    Mode := cmCTS;
    Init(PChar(Key)^, Length(Key), nil);
    EncodeBuffer(GetTestVector^, Data, 32);
    Result := Result and (MemCompare(TestVector, @Data, 32) = 0);
    Done;
    DecodeBuffer(Data, Data, 32);
    Result := Result and (MemCompare(GetTestVector, @Data, 32) = 0);
  finally
    CheckCipherKeySize := SaveKeyCheck;
    Free;
  end;
  FillChar(Data, SizeOf(Data), 0);
end;

procedure TCipher.Init(const Key; Size: Integer; IVector: Pointer);
begin
end;

procedure TCipher.InitKey(const Key: String; IVector: Pointer);
var
  I: Integer;
begin
  Hash.Init;
  Hash.Calc(PChar(Key)^, Length(Key));
  Hash.Done;
  I := Hash.DigestKeySize;
  if I > FKeySize then I := FKeySize;
  Init(Hash.DigestKey^, I, IVector);
  EncodeBuffer(Hash.DigestKey^, Hash.DigestKey^, Hash.DigestKeySize);
  Done;
  SetFlag(0, True);
end;

procedure TCipher.Done;
begin
  if MemCompare(FVector, FFeedback, FBufSize) = 0 then Exit;
  Move(FFeedback^, FBuffer^, FBufSize);
  Move(FVector^, FFeedback^, FBufSize);
end;

procedure TCipher.Protect;
begin
  SetFlag(0, False);
  Initialized := False;
// a Crypto Fanatican say: this is better !!
  FillChar(FVector^, FBufSize, $AA);
  FillChar(FFeedback^, FBufSize, $AA);
  FillChar(FBuffer^, FBufSize, $AA);
  FillChar(FUser^, FUserSize, $AA);

  FillChar(FVector^, FBufSize, $55);
  FillChar(FFeedback^, FBufSize, $55);
```

```
  FillChar(FBuffer^, FBufSize, $55);
  FillChar(FUser^, FUserSize, $55);

  FillChar(FVector^, FBufSize, $FF);
  FillChar(FFeedback^, FBufSize, $FF);
  FillChar(FBuffer^, FBufSize, 0);
  FillChar(FUser^, FUserSize, 0);
end;

function TCipher.GetHash: THash;
begin
  if not IsObject(FHash, THash) then
  begin
    if FHashClass = nil then FHashClass := DefaultHashClass;
    FHash := FHashClass.Create(nil);
    FHash.AddRef;
  end;
  Result := FHash;
end;

procedure TCipher.SetHashClass(Value: THashClass);
begin
  if Value <> FHashClass then
  begin
    FHash.Release;
    FHash := nil;
    FHashClass := Value;
    if FHashClass = nil then FHashClass := DefaultHashClass;
  end;
end;

procedure TCipher.InternalCodeStream(Source, Dest: TStream; DataSize: Integer;
Encode: Boolean);
const
  maxBufSize = 1024 * 4;
var
  Buf: PChar;
  SPos: Integer;
  DPos: Integer;
  Len: Integer;
  Proc: procedure(const Source; var Dest; DataSize: Integer) of object;
  Size: Integer;
begin
  if Source = nil then Exit;
  if Encode or (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) then Proc :=
EncodeBuffer
    else Proc := DecodeBuffer;
  if Dest = nil then Dest := Source;
  if DataSize < 0 then
  begin
    DataSize := Source.Size;
    Source.Position := 0;
  end;
  Buf := nil;
  Size := DataSize;
  DoProgress(Self, 0, Size);
  try
    Buf    := AllocMem(maxBufSize);
    DPos   := Dest.Position;
    SPos   := Source.Position;
    if Mode in [cmCTSMAC, cmCBCMAC, cmCFBMAC] then
    begin
      while DataSize > 0 do
      begin
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
```

```
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
    end else
      while DataSize > 0 do
      begin
        Source.Position := SPos;
        Len := DataSize;
        if Len > maxBufSize then Len := maxBufSize;
        Len := Source.Read(Buf^, Len);
        SPos := Source.Position;
        if Len <= 0 then Break;
        Proc(Buf^, Buf^, Len);
        Dest.Position := DPos;
        Dest.Write(Buf^, Len);
        DPos := Dest.Position;
        Dec(DataSize, Len);
        DoProgress(Self, Size - DataSize, Size);
      end;
  finally
    DoProgress(Self, 0, 0);
    ReallocMem(Buf, 0);
  end;
end;

procedure TCipher.InternalCodeFile(const Source, Dest: String; Encode: Boolean);
var
  S,D: TFileStream;
begin
  S := nil;
  D := nil;
  try
    if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then
    begin
      S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
      D := S;
    end else
      if (AnsiCompareText(Source, Dest) <> 0) and (Trim(Dest) <> '') then
      begin
        S := TFileStream.Create(Source, fmOpenRead or fmShareDenyNone);
        D := TFileStream.Create(Dest, fmCreate);
      end else
      begin
        S := TFileStream.Create(Source, fmOpenReadWrite);
        D := S;
      end;
    InternalCodeStream(S, D, -1, Encode);
  finally
    S.Free;
    if S <> D then
    begin
{$IFDEF VER_D3H}
      D.Size := D.Position;
{$ENDIF}
      D.Free;
    end;
  end;
end;

procedure TCipher.EncodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, True);
end;

procedure TCipher.DecodeStream(const Source, Dest: TStream; DataSize: Integer);
begin
  InternalCodeStream(Source, Dest, DataSize, False);
end;
```

```
procedure TCipher.EncodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, True);
end;

procedure TCipher.DecodeFile(const Source, Dest: String);
begin
  InternalCodeFile(Source, Dest, False);
end;

function TCipher.EncodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  EncodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

function TCipher.DecodeString(const Source: String): String;
begin
  SetLength(Result, Length(Source));
  DecodeBuffer(PChar(Source)^, PChar(Result)^, Length(Source));
  if Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC] then Result := '';
end;

procedure TCipher.EncodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Encode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        while DataSize >= FBufSize do
        begin
          XORBuffers(S, FFeedback, FBufSize, D);
          Encode(D);
          XORBuffers(D, FFeedback, FBufSize, FFeedback);
          Inc(S, FBufSize);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(S, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
```

```
cmCBC:
  begin
    F := FFeedback;
    while DataSize >= FBufSize do
    begin
      XORBuffers(S, F, FBufSize, D);
      Encode(D);
      F := D;
      Inc(S, FBufSize);
      Inc(D, FBufSize);
      Dec(DataSize, FBufSize);
    end;
    Move(F^, FFeedback^, FBufSize);
    if DataSize > 0 then
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      XORBuffers(S, FBuffer, DataSize, D);
      XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
    end;
  end;
cmCFB:
  while DataSize > 0 do
  begin
    Move(FFeedback^, FBuffer^, FBufSize);
    Encode(FBuffer);
    D^ := S^ xor PByte(FBuffer)^;
    Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
    PByteArray(FFeedback)[FBufSize-1] := D^;
    Inc(D);
    Inc(S);
    Dec(DataSize);
  end;
cmOFB:
  while DataSize > 0 do
  begin
    Move(FFeedback^, FBuffer^, FBufSize);
    Encode(FBuffer);
    D^ := S^ xor PByte(FBuffer)^;
    Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
    PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
    Inc(D);
    Inc(S);
    Dec(DataSize);
  end;
cmCTSMAC:
  begin
    while DataSize >= FBufSize do
    begin
      XORBuffers(S, FFeedback, FBufSize, FBuffer);
      Encode(FBuffer);
      XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
      Inc(S, FBufSize);
      Dec(DataSize, FBufSize);
    end;
    if DataSize > 0 then
    begin
      Move(FFeedback^, FBuffer^, FBufSize);
      Encode(FBuffer);
      XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
    end;
  end;
cmCBCMAC:
  begin
    while DataSize >= FBufSize do
    begin
      XORBuffers(S, FFeedback, FBufSize, FBuffer);
      Encode(FBuffer);
      Move(FBuffer^, FFeedback^, FBufSize);
```

```
          Inc(S, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFBMAC:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^ xor PByte(FBuffer)^;
        Inc(S);
        Dec(DataSize);
      end;
  end;
end;

procedure TCipher.DecodeBuffer(const Source; var Dest; DataSize: Integer);
var
  S,D,F,B: PByte;
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
  S := @Source;
  D := @Dest;
  case FMode of
    cmECB:
      begin
        if S <> D then Move(S^, D^, DataSize);
        while DataSize >= FBufSize do
        begin
          Decode(D);
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if DataSize > 0 then
        begin
          Move(D^, FBuffer^, DataSize);
          Encode(FBuffer);
          Move(FBuffer^, D^, DataSize);
        end;
      end;
    cmCTS:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          XORBuffers(D, F, FBufSize, B);
          Decode(D);
          XORBuffers(D, F, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
```

```
          Encode(FBuffer);
          XORBuffers(FBuffer, D, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCBC:
      begin
        if S <> D then Move(S^, D^, DataSize);
        F := FFeedback;
        B := FBuffer;
        while DataSize >= FBufSize do
        begin
          Move(D^, B^, FBufSize);
          Decode(D);
          XORBuffers(F, D, FBufSize, D);
          S := B;
          B := F;
          F := S;
          Inc(D, FBufSize);
          Dec(DataSize, FBufSize);
        end;
        if F <> FFeedback then Move(F^, FFeedback^, FBufSize);
        if DataSize > 0 then
        begin
          Move(FFeedback^, FBuffer^, FBufSize);
          Encode(FBuffer);
          XORBuffers(D, FBuffer, DataSize, D);
          XORBuffers(FBuffer, FFeedback, FBufSize, FFeedback);
        end;
      end;
    cmCFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := S^;
        D^ := S^ xor PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmOFB:
      while DataSize > 0 do
      begin
        Move(FFeedback^, FBuffer^, FBufSize);
        Encode(FBuffer);
        D^ := S^ xor PByte(FBuffer)^;
        Move(PByteArray(FFeedback)[1], FFeedback^, FBufSize-1);
        PByteArray(FFeedback)[FBufSize-1] := PByte(FBuffer)^;
        Inc(D);
        Inc(S);
        Dec(DataSize);
      end;
    cmCTSMAC, cmCBCMAC, cmCFBMAC:
      begin
        EncodeBuffer(Source, Dest, DataSize);
        Exit;
      end;
  end;
end;

procedure TCipher.CodeInit(Action: TPAction);
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
{  if (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) <> (Action = paCalc) then
    RaiseCipherException(errCantCalc, Format(sCantCalc, [ClassName]));}
```

```
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
  inherited CodeInit(Action);
end;

procedure TCipher.CodeDone(Action: TPAction);
begin
  inherited CodeDone(Action);
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
end;

procedure TCipher.CodeBuf(var Buffer; const BufferSize: Integer; Action:
TPAction);
begin
  if Action = paDecode then
  begin
    if Action in Actions then
      DecodeBuffer(Buffer, Buffer, BufferSize);
    inherited CodeBuf(Buffer, BufferSize, Action);
  end else
  begin
    inherited CodeBuf(Buffer, BufferSize, Action);
    if Action in Actions then
      EncodeBuffer(Buffer, Buffer, BufferSize);
  end;
end;

function TCipher.CalcMAC(Format: Integer): String;
var
  B: PByteArray;
begin
  if Mode in [cmECB, cmOFB] then
    RaiseCipherException(errInvalidMACMode, sInvalidMACMode);
  Done;
  B := AllocMem(FBufSize);
  try
    Move(FBuffer^, B^, FBufSize);
    EncodeBuffer(B^, B^, FBufSize);
    SetLength(Result, FBufSize);
    Move(FFeedback^, PChar(Result)^, FBufSize);
    if Protection <> nil then Result := Protection.CodeString(Result,
paScramble, Format)
      else Result := StrToFormat(PChar(Result), Length(Result), Format);
  finally
    ReallocMem(B, 0);
    Done;
  end;
end;

class procedure TCipher_Gost.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 32;
  AUserSize := 32;
end;

class function TCipher_Gost.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B3h,003h,0A0h,03Fh,0B5h,07Bh,091h,04Dh
         DB    097h,051h,024h,040h,0BDh,0CFh,025h,015h
         DB    034h,005h,09Ch,0F8h,0ABh,010h,086h,09Fh
         DB    0F2h,080h,047h,084h,047h,09Bh,01Ah,0D1h
end;
```

```
type
  PCipherRec = ^TCipherRec;
  TCipherRec = packed record
                  case Integer of
                    0: (X: array[0..7] of Byte);
                    1: (A, B: LongWord);
                end;

procedure TCipher_Gost.Encode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  K := User;
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := User;
    T := A + K[0];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[1];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Inc(PInteger(K), 2);
  end;
  K := @PIntArray(User)[6];
  for I := 0 to 3 do
  begin
    T := A + K[1];
    B := B xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    T := B + K[0];
    A := A xor Gost_Data[0, T        and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24        ];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Decode(Data: Pointer);
var
  I,A,B,T: LongWord;
  K: PIntArray;
begin
  A := PCipherRec(Data).A;
  B := PCipherRec(Data).B;
  K := User;
  for I := 0 to 3 do
  begin
    T := A + K[0];
    B := B xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
               Gost_Data[2, T shr 16 and $FF] xor
               Gost_Data[3, T shr 24];
    T := B + K[1];
    A := A xor Gost_Data[0, T and $FF] xor
               Gost_Data[1, T shr  8 and $FF] xor
```

```
              Gost_Data[2, T shr 16 and $FF] xor
              Gost_Data[3, T shr 24];
    Inc(PInteger(K), 2);
  end;
  for I := 0 to 11 do
  begin
    if I and 3 = 0 then K := @PIntArray(User)[6];
    T := A + K[1];
    B := B xor Gost_Data[0, T and $FF] xor
              Gost_Data[1, T shr  8 and $FF] xor
              Gost_Data[2, T shr 16 and $FF] xor
              Gost_Data[3, T shr 24];
    T := B + K[0];
    A := A xor Gost_Data[0, T and $FF] xor
              Gost_Data[1, T shr  8 and $FF] xor
              Gost_Data[2, T shr 16 and $FF] xor
              Gost_Data[3, T shr 24];
    Dec(PInteger(K), 2);
  end;
  PCipherRec(Data).A := B;
  PCipherRec(Data).B := A;
end;

procedure TCipher_Gost.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  InitEnd(IVector);
end;

class procedure TCipher_Blowfish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 56;
  AUserSize := SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key);
end;

class function TCipher_Blowfish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    019h,071h,0CAh,0CDh,02Bh,09Ch,085h,029h
         DB    0DAh,081h,047h,0B7h,0EBh,0CEh,016h,0C6h
         DB    091h,00Eh,01Dh,0C8h,040h,012h,03Eh,035h
         DB    070h,0EDh,0BCh,096h,04Ch,013h,0D0h,0B8h
end;

type
  PBlowfish = ^TBlowfish;
  TBlowfish = array[0..3, 0..255] of LongWord;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
procedure TCipher_Blowfish.Encode386(Data: Pointer);
asm  // specaly for CPU < 486
        PUSH   EDI
        PUSH   ESI
        PUSH   EBX
        PUSH   EBP
        PUSH   EDX

        MOV    ESI,[EAX].TCipher_Blowfish.FUser

        MOV    EBX,[EDX]         // A
        MOV    EDX,[EDX + 4]     // B

        XCHG   BL,BH       // here BSWAP EBX,EDX
        XCHG   DL,DH
```

```
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        XOR     EBX,[ESI + 4 * 256 * 4]
        XOR     EDI,EDI

@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP
        INC     EDI
        TEST    EDI,010h
        JZ      @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4 + 17 * 4]

        XCHG    BL,BH         // here BSWAP EBX,EDX
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;

procedure TCipher_Blowfish.Decode386(Data: Pointer);
asm // specaly for CPU < 486
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser

        MOV     EBX,[EDX]         // A
        MOV     EDX,[EDX + 4]     // B

        XCHG    BL,BH
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        XOR     EBX,[ESI + 4 * 256 * 4 + 17 * 4]
```

```
        MOV     EDI,16

@@1:    MOV     EAX,EBX
        SHR     EBX,16

        MOVZX   ECX,BH
        MOV     EBP,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 1]

        MOVZX   ECX,AH
        XOR     EBP,[ESI + ECX * 4 + 1024 * 2]
        MOVZX   ECX,AL
        ADD     EBP,[ESI + ECX * 4 + 1024 * 3]
        XOR     EDX,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR     EBP,EDX
        MOV     EDX,EAX
        MOV     EBX,EBP

        DEC     EDI
        JNZ     @@1

        POP     EAX
        XOR     EDX,[ESI + 4 * 256 * 4]

        XCHG    BL,BH         // BSWAP
        XCHG    DL,DH
        ROL     EBX,16
        ROL     EDX,16
        XCHG    BL,BH
        XCHG    DL,DH

        MOV     [EAX],EDX
        MOV     [EAX + 4],EBX

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;
  {$ENDIF} //486GE
{$ENDIF}

procedure TCipher_Blowfish.Encode(Data: Pointer);
{$IFDEF UseASM}  // спеціально для CPU >= 486
asm
        PUSH    EDI
        PUSH    ESI
        PUSH    EBX
        PUSH    EBP
        PUSH    EDX

        MOV     ESI,[EAX].TCipher_Blowfish.FUser
        MOV     EBX,[EDX]          // A
        MOV     EBP,[EDX + 4]      // B

        BSWAP   EBX                // CPU >= 486
        BSWAP   EBP

        XOR     EDI,EDI
        XOR     EBX,[ESI + 4 * 256 * 4]
//      XOR     ECX,ECX
@@1:

        MOV     EAX,EBX
        SHR     EBX,16
        MOVZX   ECX,BH     // it's faster with AMD Chips,
```

```
//        MOV    CL,BH    // it's faster with PII's
        MOV    EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX  ECX,BL
//        MOV    CL,BL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX  ECX,AH
//        MOV    CL,AH
        XOR    EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX  ECX,AL
//        MOV    CL,AL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR    EBP,[ESI + 4 * 256 * 4 + 4 + EDI * 4]

        INC    EDI
        XOR    EDX,EBP
        TEST   EDI,010h
        MOV    EBP,EAX
        MOV    EBX,EDX
        JZ     @@1

        POP    EAX
        XOR    EBP,[ESI + 4 * 256 * 4 + 17 * 4]

        BSWAP  EBX
        BSWAP  EBP

        MOV    [EAX],EBP
        MOV    [EAX + 4],EBX

        POP    EBP
        POP    EBX
        POP    ESI
        POP    EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  A := SwapInteger(PCipherRec(Data).A) xor P[0]; Inc(PInteger(P));
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    B := B xor P[0] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
                         D[3, A        and $FF]);

    A := A xor P[1] xor (D[0, B shr 24        ] +
                         D[1, B shr 16 and $FF] xor
                         D[2, B shr  8 and $FF] +
                         D[3, B        and $FF]);
    Inc(PInteger(P), 2);
  end;
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
        PUSH   EDI
        PUSH   ESI
        PUSH   EBX
```

```
        PUSH   EBP
        PUSH   EDX

        MOV    ESI,[EAX].TCipher_Blowfish.FUser
        MOV    EBX,[EDX]         // A
        MOV    EBP,[EDX + 4]     // B

        BSWAP  EBX
        BSWAP  EBP

        XOR    EBX,[ESI + 4 * 256 * 4 + 17 * 4]
        MOV    EDI,16
//        XOR    ECX,ECX

@@1:    MOV    EAX,EBX
        SHR    EBX,16

        MOVZX  ECX,BH
//        MOV    CL,BH
        MOV    EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX  ECX,BL
//        MOV    CL,BL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 1]

        MOVZX  ECX,AH
//        MOV    CL,AH
        XOR    EDX,[ESI + ECX * 4 + 1024 * 2]
        MOVZX  ECX,AL
//        MOV    CL,AL
        ADD    EDX,[ESI + ECX * 4 + 1024 * 3]
        XOR    EBP,[ESI + 4 * 256 * 4 + EDI * 4]

        XOR    EDX,EBP
        DEC    EDI
        MOV    EBP,EAX
        MOV    EBX,EDX
        JNZ    @@1

        POP    EAX
        XOR    EBP,[ESI + 4 * 256 * 4]

        BSWAP  EBX
        BSWAP  EBP

        MOV    [EAX],EBP
        MOV    [EAX + 4],EBX

        POP    EBP
        POP    EBX
        POP    ESI
        POP    EDI
end;
{$ELSE}
var
  I,A,B: LongWord;
  P: PIntArray;
  D: PBlowfish;
begin
  D := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data) + SizeOf(Blowfish_Key) -
SizeOf(Integer));
  A := SwapInteger(PCipherRec(Data).A) xor P[0];
  B := SwapInteger(PCipherRec(Data).B);
  for I := 0 to 7 do
  begin
    Dec(PInteger(P), 2);
    B := B xor P[1] xor (D[0, A shr 24        ] +
                         D[1, A shr 16 and $FF] xor
                         D[2, A shr  8 and $FF] +
```

```
                          D[3, A        and $FF]);
    A := A xor P[0] xor (D[0, B shr 24        ] +
                          D[1, B shr 16 and $FF] xor
                          D[2, B shr  8 and $FF] +
                          D[3, B        and $FF]);
  end;
  Dec(PInteger(P));
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

procedure TCipher_Blowfish.Init(const Key; Size: Integer; IVector: Pointer);
var
  I,J: Integer;
  B: array[0..7] of Byte;
  K: PByteArray;
  P: PIntArray;
  S: PBlowfish;
begin
  InitBegin(Size);
  K := @Key;
  S := User;
  P := Pointer(PChar(User) + SizeOf(Blowfish_Data));
  Move(Blowfish_Data, S^, SizeOf(Blowfish_Data));
  Move(Blowfish_Key, P^, Sizeof(Blowfish_Key));
  J := 0;
  for I := 0 to 17 do
  begin
    P[I] := P[I] xor (K[(J + 0) mod Size] shl 24 +
                      K[(J + 1) mod Size] shl 16 +
                      K[(J + 2) mod Size] shl  8 +
                      K[(J + 3) mod Size]);
    J := (J + 4) mod Size;
  end;
  FillChar(B, SizeOf(B), 0);
  for I := 0 to 8 do
  begin
    Encode(@B);
    P[I * 2]     := SwapInteger(PCipherRec(@B).A);
    P[I * 2 + 1] := SwapInteger(PCipherRec(@B).B);
  end;
  for I := 0 to 3 do
    for J := 0 to 127 do
    begin
      Encode(@B);
      S[I, J * 2]   := SwapInteger(PCipherRec(@B).A);
      S[I, J * 2 +1] := SwapInteger(PCipherRec(@B).B);
    end;

  FillChar(B, SizeOf(B), 0);
  InitEnd(IVector);
end;

class procedure TCipher_IDEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 208;
end;

class function TCipher_IDEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,065h,0CAh,0D8h,043h,0E7h,099h,093h
         DB    0EDh,041h,0EAh,048h,0FDh,066h,050h,094h
         DB    0A2h,025h,06Dh,0D7h,0B1h,0D0h,09Ah,023h
```

```
          DB    03Dh,0D2h,0E8h,0ECh,0C9h,045h,07Fh,07Eh
end;

function IDEAMul(X, Y: LongWord): LongWord; assembler; register;
asm
     AND    EAX,0FFFFh
     JZ     @@1
     AND    EDX,0FFFFh
     JZ     @@1
     MUL    EDX
     MOV    ECX,EAX
     MOV    EDX,EAX
     SHR    EDX,16
     SUB    EAX,EDX
     CMP    AX,CX
     JNA    @@2
     INC    EAX
@@2: RET
@@1: MOV    ECX,1
     SUB    ECX,EAX
     SUB    ECX,EDX
     MOV    EAX,ECX
end;

procedure TCipher_IDEA.Cipher(Data, Key: PWordArray);
var
  I: LongWord;
  X,Y,A,B,C,D: LongWord;
begin
  I := SwapInteger(PIntArray(Data)[0]);
  A := LongRec(I).Hi;
  B := LongRec(I).Lo;
  I := SwapInteger(PIntArray(Data)[1]);
  C := LongRec(I).Hi;
  D := LongRec(I).Lo;
  for I := 0 to 7 do
  begin
    A := IDEAMul(A, Key[0]);
    Inc(B, Key[1]);
    Inc(C, Key[2]);
    D := IDEAMul(D, Key[3]);
    Y := C xor A;
    Y := IDEAMul(Y, Key[4]);
    X := B xor D + Y;
    X := IDEAMul(X, Key[5]);
    Inc(Y, X);
    A := A xor X;
    D := D xor Y;
    Y := B xor Y;
    B := C xor X;
    C := Y;
    Inc(PWord(Key), 6);
  end;
  LongRec(I).Hi := IDEAMul(A, Key[0]);
  LongRec(I).Lo := C + Key[1];
  PIntArray(Data)[0] := SwapInteger(I);
  LongRec(I).Hi := B + Key[2];
  LongRec(I).Lo := IDEAMul(D, Key[3]);
  PIntArray(Data)[1] := SwapInteger(I);
end;

procedure TCipher_IDEA.Encode(Data: Pointer);
begin
  Cipher(Data, User);
end;

procedure TCipher_IDEA.Decode(Data: Pointer);
begin
  Cipher(Data, @PIntArray(User)[26]);
```

```
end;

procedure TCipher_IDEA.Init(const Key; Size: Integer; IVector: Pointer);

  function IDEAInv(X: Word): Word;
  var
    A, B, C, D: Word;
  begin
    if X <= 1 then
    begin
      Result := X;
      Exit;
    end;
    A := 1;
    B := $10001 div X;
    C := $10001 mod X;
    while C <> 1 do
    begin
      D := X div C;
      X := X mod C;
      Inc(A, B * D);
      if X = 1 then
      begin
        Result := A;
        Exit;
      end;
      D := C div X;
      C := C mod X;
      Inc(B, A * D);
    end;
    Result := 1 - B;
  end;

var
  I: Integer;
  E: PWordArray;
  A,B,C: Word;
  K,D: PWordArray;
begin
  InitBegin(Size);
  E := User;
  Move(Key, E^, Size);
  for I := 0 to 7 do E[I] := Swap(E[I]);
  for I := 0 to 39 do
    E[I + 8] := E[I and not 7 + (I + 1) and 7] shl 9 or
                E[I and not 7 + (I + 2) and 7] shr 7;
  for I := 41 to 44 do
    E[I + 7] := E[I] shl 9 or E[I + 1] shr 7;
  K  := E;
  D  := @E[100];
  A  := IDEAInv(K[0]);
  B  := 0 - K[1];
  C  := 0 - K[2];
  D[3] := IDEAInv(K[3]);
  D[2] := C;
  D[1] := B;
  D[0] := A;
  Inc(PWord(K), 4);
  for I := 1 to 8 do
  begin
    Dec(PWord(D), 6);
    A    := K[0];
    D[5] := K[1];
    D[4] := A;
    A    := IDEAInv(K[2]);
    B    := 0 - K[3];
    C    := 0 - K[4];
    D[3] := IDEAInv(K[5]);
    D[2] := B;
```

```
    D[1] := C;
    D[0] := A;
    Inc(PWord(K), 6);
  end;
  A := D[2]; D[2] := D[1]; D[1] := A;
  InitEnd(IVector);
end;

type
  PSAFERRec = ^TSAFERRec;
  TSAFERRec = packed record
                case Integer of
                  0: (A,B,C,D,E,F,G,H: Byte);
                  1: (X,Y: Integer);
              end;

procedure TCipher_SAFER.SetRounds(Value: Integer);
begin
  if (Value < 4) or (Value > 13) then
    case FSaferMode of
      smK40, smSK40: Value := 5;
      smK64, smSK64: Value := 6;
      smK128, smSK128: Value := 10;
    else
      Value := 8;
    end;
  FRounds := Value;
end;

class procedure TCipher_SAFER.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 768;
end;

class function TCipher_SAFER.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    000h,03Dh,049h,020h,073h,063h,085h,0AAh
         DB    0D9h,0C2h,00Ah,0DEh,07Eh,09Eh,0E9h,0ABh
         DB    024h,0D0h,074h,034h,047h,07Eh,021h,01Dh
         DB    055h,0F9h,035h,028h,098h,084h,0A8h,075h
end;

procedure TCipher_SAFER.Encode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 512);
  with PSAFERRec(Data)^ do
  begin
    for I := 1 to FRounds do
    begin
      A := A xor Key[0];
      B := B  +  Key[1];
      C := C  +  Key[2];
      D := D xor Key[3];
      E := E xor Key[4];
      F := F  +  Key[5];
      G := G  +  Key[6];
      H := H xor Key[7];
```

```
      A := Exp[A]  +  Key[8];
      B := Log[B] xor Key[9];
      C := Log[C] xor Key[10];
      D := Exp[D]  +  Key[11];
      E := Exp[E]  +  Key[12];
      F := Log[F] xor Key[13];
      G := Log[G] xor Key[14];
      H := Exp[H]  +  Key[15];

      Inc(B, A); Inc(A, B);
      Inc(D, C); Inc(C, D);
      Inc(F, E); Inc(E, F);
      Inc(H, G); Inc(G, H);

      Inc(C, A); Inc(A, C);
      Inc(G, E); Inc(E, G);
      Inc(D, B); Inc(B, D);
      Inc(H, F); Inc(F, H);

      Inc(E, A); Inc(A, E);
      Inc(F, B); Inc(B, F);
      Inc(G, C); Inc(C, G);
      Inc(H, D); Inc(D, H);

      T := B; B := E; E := C; C := T;
      T := D; D := F; F := G; G := T;

      Inc(PByte(Key), 16);
    end;
    A := A xor Key[0];
    B := B  +  Key[1];
    C := C  +  Key[2];
    D := D xor Key[3];
    E := E xor Key[4];
    F := F  +  Key[5];
    G := G  +  Key[6];
    H := H xor Key[7];
  end;
end;

procedure TCipher_SAFER.Decode(Data: Pointer);
var
  Exp,Log,Key: PByteArray;
  I: Integer;
  T: Byte;
begin
  Exp := User;
  Log := Pointer(PChar(User) + 256);
  Key := Pointer(PChar(User) + 504 + 8 * (FRounds * 2 + 1));
  with PSAFERRec(Data)^ do
  begin
    H := H xor Key[7];
    G := G  -  Key[6];
    F := F  -  Key[5];
    E := E xor Key[4];
    D := D xor Key[3];
    C := C  -  Key[2];
    B := B  -  Key[1];
    A := A xor Key[0];

    for I := 1 to FRounds do
    begin
      Dec(PByte(Key), 16);
      T := E; E := B; B := C; C := T;
      T := F; F := D; D := G; G := T;

      Dec(A, E); Dec(E, A);
      Dec(B, F); Dec(F, B);
      Dec(C, G); Dec(G, C);
```

```
      Dec(D, H); Dec(H, D);

      Dec(A, C); Dec(C, A);
      Dec(E, G); Dec(G, E);
      Dec(B, D); Dec(D, B);
      Dec(F, H); Dec(H, F);

      Dec(A, B); Dec(B, A);
      Dec(C, D); Dec(D, C);
      Dec(E, F); Dec(F, E);
      Dec(G, H); Dec(H, G);

      H := H  -  Key[15];
      G := G xor Key[14];
      F := F xor Key[13];
      E := E  -  Key[12];
      D := D  -  Key[11];
      C := C xor Key[10];
      B := B xor Key[9];
      A := A  -  Key[8];

      H := Log[H] xor Key[7];
      G := Exp[G]  -  Key[6];
      F := Exp[F]  -  Key[5];
      E := Log[E] xor Key[4];
      D := Log[D] xor Key[3];
      C := Exp[C]  -  Key[2];
      B := Exp[B]  -  Key[1];
      A := Log[A] xor Key[0];
    end;
  end;
end;

procedure TCipher_SAFER.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smStrong);
end;

procedure TCipher_SAFER.InitNew(const Key; Size: Integer; IVector: Pointer;
SAFERMode: TSAFERMode);

  procedure InitTab;
  var
    I,E: Integer;
    Exp: PByte;
    Log: PByteArray;
  begin
    Exp := User;
    Log := Pointer(PChar(User) + 256);
    E   := 1;
    for I := 0 to 255 do
    begin
      Exp^ := E and $FF;
      Log[E and $FF] := I;
      E := (E * 45) mod 257;
      Inc(Exp);
    end;
  end;

  procedure InitKey;

    function ROR3(Value: Byte): Byte; assembler;
    asm
      ROR AL,3
    end;

    function ROL6(Value: Byte): Byte; assembler;
    asm
      ROL AL,6
```

```
    end;

  var
    D: PByte;
    Exp: PByteArray;
    Strong: Boolean;
    K: array[Boolean, 0..8] of Byte;
    I,J: Integer;
  begin
    Strong := FSAFERMode in [smStrong, smSK40, smSK64, smSK128];
    Exp := User;
    D := User;
    Inc(D, 512);
    FillChar(K, SizeOf(K), 0);
{Встановлюється ключ A}
    I := Size;
    if I > 8 then I := 8;
    Move(Key, K[False], I);

    if FSAFERMode in [smK40, smSK40] then
    begin
      K[False, 5] := K[False, 0] xor K[False, 2] xor 129;
      K[False, 6] := K[False, 0] xor K[False, 3] xor K[False, 4] xor 66;
      K[False, 7] := K[False, 1] xor K[False, 2] xor K[False, 4] xor 36;
      K[False, 8] := K[False, 1] xor K[False, 3] xor 24;
      Move(K[False], K[True], SizeOf(K[False]));
    end else
    begin
      if Size > 8 then
      begin
        I := Size - 8;
        if I > 8 then I := 8;
        Move(TByteArray(Key)[8], K[True], I);
      end else Move(K[False], K[True], 9);
      for I := 0 to 7 do
      begin
        K[False, 8] := K[False, 8] xor K[False, I];
        K[True, 8]  := K[True, 8]  xor K[True, I];
      end;
    end;
{Встановлюються дані ключа}
    Move(K[True], D^, 8);
    Inc(D, 8);

    for I := 0 to 8 do K[False, I] := ROR3(K[False, I]);

    for I := 1 to FRounds do
    begin
      for J := 0 to 8 do
      begin
        K[False, J] := ROL6(K[False, J]);
        K[True, J] := ROL6(K[True, J]);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[False, (J + I * 2 -1) mod 9] + Exp[Exp[18 * I + J
+1]]
          else D^ := K[False, J] + Exp[Exp[18 * I + J +1]];
        Inc(D);
      end;
      for J := 0 to 7 do
      begin
        if Strong then D^ := K[True, (J + I * 2) mod 9] + Exp[Exp[18 * I + J
+10]]
          else D^ := K[True, J] + Exp[Exp[18 * I + J +10]];
        Inc(D);
      end;
    end;
    FillChar(K, SizeOf(K), 0);
```

```
  end;

begin
  InitBegin(Size);
  FSAFERMode := SAFERMode;
  if SAFERMode = smDefault then
    if Size <= 5 then FSAFERMode := smK40 else
      if Size <= 8 then FSAFERMode := smK64 else FSAFERMode := smK128
  else
    if SAFERMode = smStrong then
      if Size <= 5 then FSAFERMode := smSK40 else
        if Size <= 8 then FSAFERMode := smSK64 else FSAFERMode := smSK128;
  SetRounds(FRounds);
  InitTab;
  InitKey;
  InitEnd(IVector);
end;

class procedure TCipher_SAFER_K40.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 5;
end;

class function TCipher_SAFER_K40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    005h,0B4h,019h,057h,026h,05Ch,013h,060h
         DB    0A0h,082h,094h,045h,0D6h,0A5h,046h,0D8h
         DB    073h,050h,096h,080h,04Fh,06Dh,0F7h,0E5h
         DB    0C8h,01Ah,0EFh,044h,04Ch,0B4h,059h,013h
end;

procedure TCipher_SAFER_K40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK40);
end;

class function TCipher_SAFER_SK40.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0D9h,003h,003h,06Dh,018h,038h,0D1h,0C1h
         DB    089h,0E8h,038h,012h,07Fh,028h,0FCh,0C7h
         DB    0C5h,00Bh,0B7h,0C4h,0DBh,021h,0A4h,031h
         DB    020h,008h,08Ah,077h,0F7h,0DFh,026h,0FFh
end;

procedure TCipher_SAFER_SK40.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK40);
end;

class procedure TCipher_SAFER_K64.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 8;
end;

class function TCipher_SAFER_K64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,0B2h,032h,0F0h,00Eh,0C2h,0DAh,0CBh
         DB    039h,008h,02Dh,05Ch,093h,0FFh,0CEh,0F3h
         DB    08Fh,01Fh,0B7h,02Ch,0C5h,0C7h,0A7h,0E9h
```

```
        DB    089h,0BEh,061h,08Bh,000h,0E6h,09Fh,00Eh
end;

procedure TCipher_SAFER_K64.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK64);
end;

class function TCipher_SAFER_SK64.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    0DDh,09Ch,01Ah,0D6h,029h,00Ch,0EEh,04Fh
        DB    0E5h,04Bh,0C0h,055h,0BFh,022h,00Eh,0BCh
        DB    019h,041h,078h,0CFh,094h,0DBh,02Fh,039h
        DB    06Bh,01Eh,0A7h,0CAh,04Bh,05Fh,077h,0E0h
end;

procedure TCipher_SAFER_SK64.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK64);
end;

class procedure TCipher_SAFER_K128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  inherited GetContext(ABufSize, AKeySize, AUserSize);
  AKeySize := 16;
end;

class function TCipher_SAFER_K128.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    00Ch,0A9h,070h,0B9h,0F3h,014h,087h,0D9h
        DB    09Eh,05Eh,078h,031h,074h,0DFh,0A8h,0BBh
        DB    03Dh,040h,0A5h,0D9h,08Ch,07Ch,004h,0B7h
        DB    09Ch,001h,0DAh,063h,0ABh,026h,035h,0BCh
end;

procedure TCipher_SAFER_K128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smK128);
end;

class function TCipher_SAFER_SK128.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    0C8h,0A6h,070h,033h,029h,038h,038h,02Bh
        DB    069h,0ACh,061h,072h,08Fh,0DCh,09Fh,0A4h
        DB    09Eh,06Fh,0C4h,053h,0D8h,089h,0FFh,042h
        DB    072h,009h,07Dh,0CDh,0D0h,0EAh,07Eh,028h
end;

procedure TCipher_SAFER_SK128.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitNew(Key, Size, IVector, smSK128);
end;

type
  PTEARec = ^TTEARec;
  TTEARec = packed record
              A,B,C,D: LongWord;
            end;

const
  TEA_Delta = $9E3779B9;
```

```
procedure TCipher_TEA.SetRounds(Value: Integer);
begin
  FRounds := Value;
  if FRounds < 16 then FRounds := 16 else
    if FRounds > 32 then FRounds := 32;
end;

class procedure TCipher_TEA.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 32;
end;

class function TCipher_TEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0B7h,0B8h,0AAh,0BBh,026h,04Bh,006h,0F9h
         DB    070h,086h,0B0h,0E4h,056h,004h,029h,0CCh
         DB    0BFh,055h,0EAh,04Eh,0EFh,059h,026h,018h
         DB    019h,0B0h,003h,07Ch,029h,08Ch,0E2h,077h
end;

procedure TCipher_TEA.Encode(Data: Pointer);
{$IFDEF UseASM}
asm
         PUSH  EDI
         PUSH  ESI
         PUSH  EBX
         PUSH  EBP
         PUSH  EDX

         MOV   EBX,[EDX]       // X
         MOV   EDX,[EDX + 4]   // Y
         XOR   EDI,EDI         // Sum

         MOV   ESI,[EAX].TCipher_TEA.FUser  // User
         MOV   ECX,[EAX].TCipher_TEA.FRounds  // Rounds

@@1:     ADD   EDI,TEA_Delta

         MOV   EAX,EDX
         MOV   EBP,EDX
         SHL   EAX,4
         SHR   EBP,5
         ADD   EAX,[ESI]
         ADD   EBP,[ESI +  4]
         XOR   EAX,EDX
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EAX,EBX
         MOV   EBX,EAX
         SHL   EAX,4
         MOV   EBP,EBX
         SHR   EBP,5
         ADD   EAX,[ESI +  8]
         XOR   EAX,EBX
         ADD   EBP,[ESI + 12]
         ADD   EAX,EDI

         XOR   EAX,EBP
         ADD   EDX,EAX

         DEC   ECX
         JNZ   @@1
```

```
          POP   EAX
          MOV   [EAX],EBX
          MOV   [EAX + 4],EDX

          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Inc(Sum, TEA_Delta);
      Inc(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Inc(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
          PUSH  EDI
          PUSH  ESI
          PUSH  EBX
          PUSH  EBP
          PUSH  EDX

          MOV   EBX,[EDX]      // X
          MOV   EDX,[EDX + 4]  // Y

          MOV   ESI,[EAX].TCipher_TEA.FUser  // User
          MOV   EDI,TEA_Delta
          MOV   ECX,[EAX].TCipher_TEA.FRounds  // Rounds
          IMUL  EDI,ECX

@@1:      MOV   EAX,EBX
          MOV   EBP,EBX
          SHL   EAX,4
          SHR   EBP,5
          ADD   EAX,[ESI +  8]
          ADD   EBP,[ESI + 12]
          XOR   EAX,EBX
          ADD   EAX,EDI
          XOR   EAX,EBP
          SUB   EDX,EAX
          MOV   EAX,EDX
          SHL   EAX,4
          MOV   EBP,EDX
          SHR   EBP,5
          ADD   EAX,[ESI]
          XOR   EAX,EDX
          ADD   EBP,[ESI + 4]
          ADD   EAX,EDI

          XOR   EAX,EBP
          SUB   EDI,TEA_Delta
          SUB   EBX,EAX

          DEC   ECX
```

```
          JNZ   @@1

          POP   EAX
          MOV   [EAX],EBX
          MOV   [EAX + 4],EDX

          POP   EBP
          POP   EBX
          POP   ESI
          POP   EDI
end;
{$ELSE}
var
  I,Sum,X,Y: LongWord;
begin
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 + C) xor X + Sum xor (X shr 5 + D));
      Dec(X, (Y shl 4 + A) xor Y + Sum xor (Y shr 5 + B));
      Dec(Sum, TEA_Delta);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;
{$ENDIF}

procedure TCipher_TEA.Init(const Key; Size: Integer; IVector: Pointer);
begin
  InitBegin(Size);
  Move(Key, User^, Size);
  SetRounds(FRounds);
  InitEnd(IVector);
end;

class function TCipher_TEAN.TestVector: Pointer;
asm
          MOV   EAX,OFFSET @Vector
          RET
@Vector:  DB    0CDh,07Eh,0BBh,0A2h,092h,01Ah,04Bh,03Bh
          DB    0E2h,09Eh,062h,0CFh,0F7h,01Dh,0A5h,0DFh
          DB    063h,033h,094h,029h,0E2h,036h,07Ch,066h
          DB    03Fh,0F8h,01Ah,0F9h,002h,078h,0BFh,0A1h
end;

procedure TCipher_TEAN.Encode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := 0;
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  for I := 1 to FRounds do
  begin
    Inc(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    Inc(Sum, TEA_Delta);
    Inc(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
  end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

procedure TCipher_TEAN.Decode(Data: Pointer);
var
```

```
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
  K := User;
  with PTEARec(User)^ do
    for I := 1 to FRounds do
    begin
      Dec(Y, (X shl 4 xor X shr 5) + (X xor Sum) + K[Sum shr 11 and 3]);
      Dec(Sum, TEA_Delta);
      Dec(X, (Y shl 4 xor Y shr 5) + (Y xor Sum) + K[Sum and 3]);
    end;
  PTEARec(Data).A := X;
  PTEARec(Data).B := Y;
end;

const
  SCOP_SIZE = 32; {це максімум}

class procedure TCipher_SCOP.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := SCOP_SIZE * SizeOf(Integer);
  AKeySize := 48;
  AUserSize := (384 * 4 + 4 * SizeOf(Integer)) * 2;
end;

class function TCipher_SCOP.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    014h,0C0h,009h,0E8h,073h,0B6h,053h,092h
         DB    08Bh,013h,069h,0A9h,0F2h,099h,0FEh,05Eh
         DB    0EEh,03Bh,0FDh,0C1h,050h,059h,00Eh,094h
         DB    062h,017h,008h,01Eh,0A4h,01Ah,04Dh,08Fh
end;

procedure TCipher_SCOP.Encode(Data: Pointer);
var
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
  B: PInteger;
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Inc(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Decode(Data: Pointer);
var
```

```
  I, J, W: Byte;
  T, T1, T2, T3: Integer;
  P: PIntArray;
  B: PInteger;
begin
  P  := User;
  I  := P[0];
  J  := P[1];
  T3 := P[3];
  P  := @P[4 + 128];
  B  := Data;
  for W := 1 to SCOP_SIZE do
  begin
    T1 := P[J];
    Inc(J, T3);
    T := P[I - 128];
    T2 := P[J];
    Inc(I);
    T3 := T2 + T;
    P[J] := T3;
    Inc(J, T2);
    Dec(B^, T1 + T2);
    Inc(B);
  end;
end;

procedure TCipher_SCOP.Init(const Key; Size: Integer; IVector: Pointer);
var
  Init_State: packed record
                Coef: array[0..7, 0..3] of Byte;
                X: array[0..3] of LongWord;
              end;

  procedure ExpandKey;
  var
    P: PByteArray;
    I,C: Integer;
  begin
    C := 1;
    P := @Init_State;
    Move(Key, P^, Size);
    for I := Size to 47 do P[I] := P[I - Size] + P[I - Size +1];
    for I := 0 to 31 do
      if P[I] = 0 then
      begin
        P[I] := C;
        Inc(C);
      end;
  end;

  procedure GP8(Data: PIntArray);
  var
    I,I2: Integer;
    NewX: array[0..3] of LongWord;
    X1,X2,X3,X4: LongWord;
    Y1,Y2: LongWord;
  begin
    I := 0;
    while I < 8 do
    begin
      I2 := I shr 1;
      X1 := Init_State.X[I2] shr 16;
      X2 := X1 * X1;
      X3 := X2 * X1;
      X4 := X3 * X1;
      Y1 := Init_State.Coef[I][0] * X4 +
            Init_State.Coef[I][1] * X3 +
            Init_State.Coef[I][2] * X2 +
            Init_State.Coef[I][3] * X1 + 1;
```

```
      X1 := Init_State.X[I2] and $FFFF;
      X2 := X1 * X1;
      X3 := X2 * X1;
      X4 := X3 * X1;
      Y2 := Init_State.Coef[I +1][0] * X4 +
            Init_State.Coef[I +2][1] * X3 +
            Init_State.Coef[I +3][2] * X2 +
            Init_State.Coef[I +4][3] * X1 + 1;
      Data[I2] := Y1 shl 16 or Y2 and $FFFF;
      NewX[I2] := Y1 and $FFFF0000 or Y2 shr 16;
      Inc(I, 2);
    end;
    Init_State.X[0] := NewX[0] shr 16 or NewX[3] shl 16;
    Init_State.X[1] := NewX[0] shl 16 or NewX[1] shr 16;
    Init_State.X[2] := NewX[1] shl 16 or NewX[2] shr 16;
    Init_State.X[3] := NewX[2] shl 16 or NewX[3] shr 16;
  end;

var
  I,J: Integer;
  T: array[0..3] of Integer;
  P: PIntArray;
begin
  InitBegin(Size);
  FillChar(Init_State, SizeOf(Init_State), 0);
  FillChar(T, SizeOf(T), 0);
  P := Pointer(PChar(User) + 12);
  ExpandKey;
  for I := 0 to 7 do GP8(@T);
  for I := 0 to 11 do
  begin
    for J := 0 to 7 do GP8(@P[I * 32 + J * 4]);
    GP8(@T);
  end;
  GP8(@T);
  I := T[3] and $7F;
  P[I] := P[I] or 1;
  P := User;
  P[0] := T[3] shr 24;
  P[1] := T[3] shr 16;
  P[2] := T[3] shr  8;
  FillChar(Init_State, SizeOf(Init_State), 0);
  InitEnd(IVector);
  P := Pointer(PChar(User) + FUserSize shr 1);
  Move(User^, P^, FUserSize shr 1);
end;

procedure TCipher_SCOP.Done;
begin
  inherited Done;
  Move(PByteArray(User)[FUserSize shr 1], User^, FUserSize shr 1);
end;

class procedure TCipher_Q128.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 256;
end;

class function TCipher_Q128.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    099h,0AAh,0D0h,03Dh,0CAh,014h,04Eh,02Ah
         DB    0F8h,01Eh,001h,0A0h,0EAh,0ABh,09Fh,048h
         DB    023h,02Dh,059h,054h,054h,07Eh,02Bh,012h
         DB    086h,080h,0E8h,033h,0EBh,0E1h,05Eh,0AEh
```

```
end;

procedure TCipher_Q128.Encode(Data: Pointer);
{$IFDEF UseASM}
asm
       PUSH   ESI
       PUSH   EDI
       PUSH   EBX
       PUSH   EBP
       PUSH   EDX

       MOV    EDI,[EAX].TCipher_Q128.FUser

       MOV    EAX,[EDX]       // B0
       MOV    EBX,[EDX +  4]  // B1
       MOV    ECX,[EDX +  8]  // B2
       MOV    EDX,[EDX + 12]  // B3

       MOV    EBP,16

@@1:   MOV    ESI,EAX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ROL    ESI,10
       ADD    EAX,[EDI]
       XOR    EAX,EBX

       MOV    EBX,EAX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ROL    EBX,10
       ADD    EAX,[EDI + 4]
       XOR    EAX,ECX

       MOV    ECX,EAX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ROL    ECX,10
       ADD    EAX,[EDI + 8]
       XOR    EAX,EDX

       MOV    EDX,EAX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ROL    EDX,10
       ADD    EAX,[EDI + 12]
       XOR    EAX,ESI

       ADD    EDI,16

       DEC    EBP
       JNZ    @@1

       POP    ESI

       MOV    [ESI],EAX       // B0
       MOV    [ESI +  4],EBX  // B1
       MOV    [ESI +  8],ECX  // B2
       MOV    [ESI + 12],EDX  // B3


       POP    EBP
       POP    EBX
       POP    EDI
       POP    ESI
end;
{$ELSE}
var
  D: PInteger;
```

```
  B0, B1, B2, B3, I: LongWord;
begin
  D  := User;
  B0 := PIntArray(Data)[0];
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B1 := B1 xor (Q128_Data[B0 and $03FF] + D^); Inc(D); B0 := B0 shl 10 or B0
shr 22;
    B2 := B2 xor (Q128_Data[B1 and $03FF] + D^); Inc(D); B1 := B1 shl 10 or B1
shr 22;
    B3 := B3 xor (Q128_Data[B2 and $03FF] + D^); Inc(D); B2 := B2 shl 10 or B2
shr 22;
    B0 := B0 xor (Q128_Data[B3 and $03FF] + D^); Inc(D); B3 := B3 shl 10 or B3
shr 22;
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}
procedure TCipher_Q128.Decode(Data: Pointer);
{$IFDEF UseASM}
asm
       PUSH   ESI
       PUSH   EDI
       PUSH   EBX
       PUSH   EBP
       PUSH   EDX

       MOV    EDI,[EAX].TCipher_Q128.FUser
       LEA    EDI,[EDI + 64 * 4]

       MOV    ESI,[EDX]      // B0
       MOV    EBX,[EDX +  4]  // B1
       MOV    ECX,[EDX +  8]  // B2
       MOV    EDX,[EDX + 12]  // B3

       MOV    EBP,16

@@1:   SUB    EDI,16

       ROR    EDX,10
       MOV    EAX,EDX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI + 12]
       XOR    ESI,EAX

       ROR    ECX,10
       MOV    EAX,ECX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  8]
       XOR    EDX,EAX

       ROR    EBX,10
       MOV    EAX,EBX
       AND    EAX,03FFh
       MOV    EAX,[EAX * 4 + OFFSET Q128_DATA]
       ADD    EAX,[EDI +  4]
       XOR    ECX,EAX

       ROR    ESI,10
       MOV    EAX,ESI
       AND    EAX,03FFh
```

```
        MOV     EAX,[EAX * 4 + OFFSET Q128_DATA]
        ADD     EAX,[EDI]
        XOR     EBX,EAX

        DEC     EBP
        JNZ     @@1

        POP     EAX

        MOV     [EAX],ESI       // B0
        MOV     [EAX +  4],EBX  // B1
        MOV     [EAX +  8],ECX  // B2
        MOV     [EAX + 12],EDX  // B3


        POP     EBP
        POP     EBX
        POP     EDI
        POP     ESI
end;
{$ELSE}
var
  D: PInteger;
  B0, B1, B2, B3, I: LongWord;
begin
  D  := @PIntArray(User)[63];
  B0 := PIntArray(Data)[0];
  B1 := PIntArray(Data)[1];
  B2 := PIntArray(Data)[2];
  B3 := PIntArray(Data)[3];
  for I := 1 to 16 do
  begin
    B3 := B3 shr 10 or B3 shl 22; B0 := B0 xor (Q128_Data[B3 and $03FF] + D^);
Dec(D);
    B2 := B2 shr 10 or B2 shl 22; B3 := B3 xor (Q128_Data[B2 and $03FF] + D^);
Dec(D);
    B1 := B1 shr 10 or B1 shl 22; B2 := B2 xor (Q128_Data[B1 and $03FF] + D^);
Dec(D);
    B0 := B0 shr 10 or B0 shl 22; B1 := B1 xor (Q128_Data[B0 and $03FF] + D^);
Dec(D);
  end;
  PIntArray(Data)[0] := B0;
  PIntArray(Data)[1] := B1;
  PIntArray(Data)[2] := B2;
  PIntArray(Data)[3] := B3;
end;
{$ENDIF}

procedure TCipher_Q128.Init(const Key; Size: Integer; IVector: Pointer);
var
  K: array[0..3] of LongWord;
  I: Integer;
  D: PInteger;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  D := User;
  for I := 19 downto 1 do
  begin
    K[1] := K[1] xor Q128_Data[K[0] and $03FF]; K[0] := K[0] shr 10 or K[0] shl
22;
    K[2] := K[2] xor Q128_Data[K[1] and $03FF]; K[1] := K[1] shr 10 or K[1] shl
22;
    K[3] := K[3] xor Q128_Data[K[2] and $03FF]; K[2] := K[2] shr 10 or K[2] shl
22;
    K[0] := K[0] xor Q128_Data[K[3] and $03FF]; K[3] := K[3] shr 10 or K[3] shl
22;
    if I <= 16 then
```

```
    begin
      D^ := K[0]; Inc(D);
      D^ := K[1]; Inc(D);
      D^ := K[2]; Inc(D);
      D^ := K[3]; Inc(D);
    end;
  end;
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

type
  P3Way_Key = ^T3Way_Key;
  T3Way_Key = packed record
                E_Key: array[0..2] of Integer;
                E_Data: array[0..11] of Integer;
                D_Key: array[0..2] of Integer;
                D_Data: array[0..11] of Integer;
              end;

class procedure TCipher_3Way.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 12;
  AKeySize := 12;
  AUserSize := SizeOf(T3Way_Key);
end;

class function TCipher_3Way.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    077h,0FCh,077h,094h,07Ch,08Fh,0DEh,021h
         DB    0E9h,081h,0DFh,02Ah,0B1h,0BCh,07Eh,0F8h
         DB    0A3h,0B6h,044h,04Bh,0B6h,0FCh,079h,0C4h
         DB    09Bh,068h,04Fh,009h,0C7h,0BFh,00Eh,005h
end;

procedure TCipher_3Way.Encode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := E_Key[0];
    K1 := E_Key[1];
    K2 := E_Key[2];
    E  := @E_Data;
  end;
  A0 := PIntArray(Data)[0];
  A1 := PIntArray(Data)[1];
  A2 := PIntArray(Data)[2];
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
```

```
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    asm
      ROR  B0,10
      ROL  B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  PIntArray(Data)[0] := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl
16 xor
                              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl
24 xor
                              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl
8;
  PIntArray(Data)[1] := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl
16 xor
                              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl
24 xor
                              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl
8;
  PIntArray(Data)[2] := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl
16 xor
                              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl
24 xor
                              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl
8;
end;

procedure TCipher_3Way.Decode(Data: Pointer);
var
  I: Integer;
  A0,A1,A2: LongWord;
  B0,B1,B2: LongWord;
  K0,K1,K2: LongWord;
  E: PLongWord;
begin
  with P3Way_Key(User)^ do
  begin
    K0 := D_Key[0];
    K1 := D_Key[1];
    K2 := D_Key[2];
    E  := @D_Data;
  end;
  A0 := SwapBits(PIntArray(Data)[2]);
  A1 := SwapBits(PIntArray(Data)[1]);
  A2 := SwapBits(PIntArray(Data)[0]);
  for I := 0 to 10 do
  begin
    A0 := A0 xor K0 xor E^ shl 16;
    A1 := A1 xor K1;
    A2 := A2 xor K2 xor E^;
    Inc(E);

    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
```

```
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
    asm
      ROR B0,10
      ROL B2,1
    end;
    A0 := B0 xor (B1 or not B2);
    A1 := B1 xor (B2 or not B0);
    A2 := B2 xor (B0 or not B1);
    asm
      ROL A0,1
      ROR A2,10
    end;
  end;
  A0 := A0 xor K0 xor E^ shl 16;
  A1 := A1 xor K1;
  A2 := A2 xor K2 xor E^;
  B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
              A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
              A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
  B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
              A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
              A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
  B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
              A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
              A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;

  PIntArray(Data)[2] := SwapBits(B0);
  PIntArray(Data)[1] := SwapBits(B1);
  PIntArray(Data)[0] := SwapBits(B2);
end;

procedure TCipher_3Way.Init(const Key; Size: Integer; IVector: Pointer);

  procedure RANDGenerate(Start: Integer; var P: Array of Integer);
  var
    I: Integer;
  begin
    for I := 0 to 11 do
    begin
      P[I] := Start;
      Start := Start shl 1;
      if Start and $10000 <> 0 then Start := Start xor $11011;
    end;
  end;

var
  A0, A1, A2: Integer;
  B0, B1, B2: Integer;
begin
  InitBegin(Size);
  with P3Way_Key(User)^ do
  begin
    Move(Key, E_Key, Size);
    Move(Key, D_Key, Size);
    RANDGenerate($0B0B, E_Data);
    RANDGenerate($B1B1, D_Data);

    A0 := D_Key[0]; A1 := D_Key[1]; A2 := D_Key[2];
    B0 := A0 xor A0 shr 16 xor A1 shl 16 xor A1 shr 16 xor A2 shl 16 xor
                A1 shr 24 xor A2 shl  8 xor A2 shr  8 xor A0 shl 24 xor
                A2 shr 16 xor A0 shl 16 xor A2 shr 24 xor A0 shl  8;
    B1 := A1 xor A1 shr 16 xor A2 shl 16 xor A2 shr 16 xor A0 shl 16 xor
                A2 shr 24 xor A0 shl  8 xor A0 shr  8 xor A1 shl 24 xor
                A0 shr 16 xor A1 shl 16 xor A0 shr 24 xor A1 shl  8;
    B2 := A2 xor A2 shr 16 xor A0 shl 16 xor A0 shr 16 xor A1 shl 16 xor
                A0 shr 24 xor A1 shl  8 xor A1 shr  8 xor A2 shl 24 xor
                A1 shr 16 xor A2 shl 16 xor A1 shr 24 xor A2 shl  8;
```

```
    D_Key[2] := SwapBits(B0); D_Key[1] := SwapBits(B1); D_Key[0] :=
SwapBits(B2);
  end;
  InitEnd(IVector);
end;

class procedure TCipher_Twofish.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 32;
  AUserSize := 4256;
end;

class function TCipher_Twofish.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    0A5h,053h,057h,003h,0EFh,033h,048h,079h
         DB    09Fh,022h,0B4h,054h,097h,005h,084h,019h
         DB    087h,0BDh,083h,01Ch,04Dh,0AEh,012h,013h
         DB    060h,07Ch,07Ch,0D1h,098h,045h,002h,019h
end;

type
  PTwofishBox = ^TTwofishBox;
  TTwofishBox = array[0..3, 0..255] of Longword;

  TLongRec = record
               case Integer of
                 0: (L: Longword);
                 1: (A,B,C,D: Byte);
             end;

procedure TCipher_Twofish.Encode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S   := User;
  A.L := PIntArray(Data)[0] xor S[0];
  B.L := PIntArray(Data)[1] xor S[1];
  C.L := PIntArray(Data)[2] xor S[2];
  D.L := PIntArray(Data)[3] xor S[3];

  S   := @PIntArray(User)[8];
  Box := @PIntArray(User)[40];
  for I := 0 to 7 do
  begin
    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C] xor Box[0, B.D];
    asm ROL  D.L,1 end;
    C.L := C.L xor (X + Y       + S[0]);
    D.L := D.L xor (X + Y shl 1 + S[1]);
    asm ROR  C.L,1 end;

    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C] xor Box[0, D.D];
    asm ROL  B.L,1 end;
    A.L := A.L xor (X + Y       + S[2]);
    B.L := B.L xor (X + Y shl 1 + S[3]);
    asm ROR  A.L,1 end;
    Inc(PInteger(S), 4);
  end;
  S := User;
  PIntArray(Data)[0] := C.L xor S[4];
  PIntArray(Data)[1] := D.L xor S[5];
```

```
  PIntArray(Data)[2] := A.L xor S[6];
  PIntArray(Data)[3] := B.L xor S[7];
end;

procedure TCipher_Twofish.Decode(Data: Pointer);
var
  S: PIntArray;
  Box: PTwofishBox;
  I,X,Y: LongWord;
  A,B,C,D: TLongRec;
begin
  S := User;
  Box := @PIntArray(User)[40];
  C.L := PIntArray(Data)[0] xor S[4];
  D.L := PIntArray(Data)[1] xor S[5];
  A.L := PIntArray(Data)[2] xor S[6];
  B.L := PIntArray(Data)[3] xor S[7];
  S := @PIntArray(User)[36];
  for I := 0 to 7 do
  begin
    X := Box[0, C.A] xor Box[1, C.B] xor Box[2, C.C] xor Box[3, C.D];
    Y := Box[0, D.D] xor Box[1, D.A] xor Box[2, D.B] xor Box[3, D.C];
    asm ROL  A.L,1 end;
    B.L := B.L xor (X + Y shl 1 + S[3]);
    A.L := A.L xor (X + Y       + S[2]);
    asm ROR  B.L,1 end;

    X := Box[0, A.A] xor Box[1, A.B] xor Box[2, A.C] xor Box[3, A.D];
    Y := Box[0, B.D] xor Box[1, B.A] xor Box[2, B.B] xor Box[3, B.C];
    asm ROL  C.L,1 end;
    D.L := D.L xor (X + Y shl 1 + S[1]);
    C.L := C.L xor (X + Y       + S[0]);
    asm ROR  D.L,1 end;
    Dec(PByte(S),16);
  end;
  S := User;
  PIntArray(Data)[0] := A.L xor S[0];
  PIntArray(Data)[1] := B.L xor S[1];
  PIntArray(Data)[2] := C.L xor S[2];
  PIntArray(Data)[3] := D.L xor S[3];
end;

procedure TCipher_Twofish.Init(const Key; Size: Integer; IVector: Pointer);
var
  BoxKey: array[0..3] of TLongRec;
  SubKey: PIntArray;
  Box: PTwofishBox;

  procedure SetupKey;

    function Encode(K0, K1: Integer): Integer;
    var
      R, I, J, G2, G3: Integer;
      B: byte;
    begin
      R := 0;
      for I := 0 to 1 do
      begin
        if I <> 0 then R := R xor K0 else R := R xor K1;
        for J := 0 to 3 do
        begin
          B := R shr 24;
          if B and $80 <> 0 then G2 := (B shl 1 xor $014D) and $FF
            else G2 := B shl 1 and $FF;
          if B and 1 <> 0 then G3 := (B shr 1 and $7F) xor $014D shr 1 xor G2
            else G3 := (B shr 1 and $7F) xor G2;
          R := R shl 8 xor G3 shl 24 xor G2 shl 16 xor G3 shl 8 xor B;
        end;
      end;
```

```
    Result := R;
  end;

  function F32(X: Integer; K: array of Integer): Integer;
  var
    A, B, C, D: Integer;
  begin
    A := X and $FF;
    B := X shr  8 and $FF;
    C := X shr 16 and $FF;
    D := X shr 24;
    if Size = 32 then
    begin
      A := Twofish_8x8[1, A] xor K[3] and $FF;
      B := Twofish_8x8[0, B] xor K[3] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[3] shr 16 and $FF;
      D := Twofish_8x8[1, D] xor K[3] shr 24;
    end;
    if Size >= 24 then
    begin
      A := Twofish_8x8[1, A] xor K[2] and $FF;
      B := Twofish_8x8[1, B] xor K[2] shr  8 and $FF;
      C := Twofish_8x8[0, C] xor K[2] shr 16 and $FF;
      D := Twofish_8x8[0, D] xor K[2] shr 24;
    end;
    A := Twofish_8x8[0, A] xor K[1] and $FF;
    B := Twofish_8x8[1, B] xor K[1] shr  8 and $FF;
    C := Twofish_8x8[0, C] xor K[1] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[1] shr 24;

    A := Twofish_8x8[0, A] xor K[0] and $FF;
    B := Twofish_8x8[0, B] xor K[0] shr  8 and $FF;
    C := Twofish_8x8[1, C] xor K[0] shr 16 and $FF;
    D := Twofish_8x8[1, D] xor K[0] shr 24;

    Result := Twofish_Data[0, A] xor Twofish_Data[1, B] xor
              Twofish_Data[2, C] xor Twofish_Data[3, D];
  end;

var
  I,J,A,B: Integer;
  E,O: array[0..3] of Integer;
  K: array[0..7] of Integer;
begin
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  if Size <= 16 then Size := 16 else
    if Size <= 24 then Size := 24
      else Size := 32;
  J := Size shr 3 - 1;
  for I := 0 to J do
  begin
    E[I] := K[I shl 1];
    O[I] := K[I shl 1 + 1];
    BoxKey[J].L := Encode(E[I], O[I]);
    Dec(J);
  end;
  J := 0;
  for I := 0 to 19 do
  begin
    A := F32(J, E);
    B := ROL(F32(J + $01010101, O), 8);
    SubKey[I shl 1] := A + B;
    B := A + B shr 1;
    SubKey[I shl 1 + 1] := ROL(B, 9);
    Inc(J, $02020202);
  end;
end;
```

```
  procedure DoXOR(D, S: PIntArray; Value: LongWord);
  var
    I: LongWord;
  begin
    Value := (Value and $FF) * $01010101;
    for I := 0 to 63 do D[I] := S[I] xor Value;
  end;

  procedure SetupBox128;
  var
    L: array[0..255] of Byte;
    A,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L);
    A := BoxKey[0].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 8);
    A := BoxKey[0].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[1].L shr 16);
    A := BoxKey[0].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, L[I]] xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[1].L shr 24);
    A := BoxKey[0].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, L[I]] xor A];
  end;

  procedure SetupBox192;
  var
    L: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L);
    A := BoxKey[0].A;
    B := BoxKey[1].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[1], BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@L, @Twofish_8x8[0], BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@L ,@Twofish_8x8[0], BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

  procedure SetupBox256;
  var
    L: array[0..255] of Byte;
    K: array[0..255] of Byte;
    A,B,I: Integer;
  begin
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L);
```

```
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L);
    A := BoxKey[0].A;
    B := BoxKey[1].A;
    for I := 0 to 255 do
      Box[0, I] := Twofish_Data[0, Twofish_8x8[0, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0], BoxKey[3].L shr 8);
    for I := 0 to 255 do L[I] := Twofish_8x8[1, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 8);
    A := BoxKey[0].B;
    B := BoxKey[1].B;
    for I := 0 to 255 do
      Box[1, I] := Twofish_Data[1, Twofish_8x8[0, Twofish_8x8[1, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[0],BoxKey[3].L shr 16);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 16);
    A := BoxKey[0].C;
    B := BoxKey[1].C;
    for I := 0 to 255 do
      Box[2, I] := Twofish_Data[2, Twofish_8x8[1, Twofish_8x8[0, L[I]] xor B]
xor A];
    DoXOR(@K, @Twofish_8x8[1], BoxKey[3].L shr 24);
    for I := 0 to 255 do L[I] := Twofish_8x8[0, K[I]];
    DoXOR(@L, @L, BoxKey[2].L shr 24);
    A := BoxKey[0].D;
    B := BoxKey[1].D;
    for I := 0 to 255 do
      Box[3, I] := Twofish_Data[3, Twofish_8x8[1, Twofish_8x8[1, L[I]] xor B]
xor A];
  end;

begin
  InitBegin(Size);
  SubKey := User;
  Box    := @SubKey[40];
  SetupKey;
  if Size = 16 then SetupBox128 else
    if Size = 24 then SetupBox192
      else SetupBox256;
  InitEnd(IVector);
end;

class procedure TCipher_Shark.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 8;
  AKeySize := 16;
  AUserSize := 112;
end;

class function TCipher_Shark.TestVector: Pointer;
asm
        MOV   EAX,OFFSET @Vector
        RET
@Vector: DB    0D9h,065h,021h,0AAh,0C0h,0C3h,084h,060h
        DB    09Dh,0CEh,01Fh,08Bh,0FBh,0ABh,018h,03Fh
        DB    0A1h,021h,0ACh,0F8h,053h,049h,0C0h,06Fh
        DB    027h,03Ah,089h,015h,0D3h,07Ah,0E9h,00Bh
end;

{$IFDEF VER_D4H} // >= D4
  {.$DEFINE Shark64}
{$ENDIF}

type
  PInt64        = ^TInt64;
{$IFDEF Shark64}
```

```
  TInt64        = Int64;
{$ELSE}
  TInt64        = packed record
                    L,R: Integer;
                  end;
{$ENDIF}

  PInt64Array = ^TInt64Array;
  TInt64Array = array[0..1023] of TInt64;

{$IFDEF Shark64}
  TShark_Data = array[0..7, 0..255] of Int64;
{$ENDIF}

procedure TCipher_Shark.Encode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  L,R: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
         TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CE)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SE[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
    Inc(PInteger(K), 2);
    T := Shark_CE[0, R shr 23 and $1FE] xor
         Shark_CE[1, R shr 15 and $1FE] xor
         Shark_CE[2, R shr  7 and $1FE] xor
         Shark_CE[3, R shl  1 and $1FE] xor
         Shark_CE[4, L shr 23 and $1FE] xor
         Shark_CE[5, L shr 15 and $1FE] xor
         Shark_CE[6, L shr  7 and $1FE] xor
         Shark_CE[7, L shl  1 and $1FE];
    R := Shark_CE[0, R shr 23 and $1FE or 1] xor
         Shark_CE[1, R shr 15 and $1FE or 1] xor
         Shark_CE[2, R shr  7 and $1FE or 1] xor
```

```
          Shark_CE[3, R shl  1 and $1FE or 1] xor
          Shark_CE[4, L shr 23 and $1FE or 1] xor
          Shark_CE[5, L shr 15 and $1FE or 1] xor
          Shark_CE[6, L shr  7 and $1FE or 1] xor
          Shark_CE[7, L shl  1 and $1FE or 1];
    L := T;
  end;
  L := L xor K[0];
  R := R xor K[1];
  Inc(PInteger(K), 2);
  L := LongWord(Shark_SE[L shr 24        ]) shl 24 xor
       LongWord(Shark_SE[L shr 16 and $FF]) shl 16 xor
       LongWord(Shark_SE[L shr  8 and $FF]) shl  8 xor
       LongWord(Shark_SE[L        and $FF]);
  R := LongWord(Shark_SE[R shr 24        ]) shl 24 xor
       LongWord(Shark_SE[R shr 16 and $FF]) shl 16 xor
       LongWord(Shark_SE[R shr  8 and $FF]) shl  8 xor
       LongWord(Shark_SE[R        and $FF]);
  PInt64(Data).L := L xor K[0];
  PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Decode(Data: Pointer);
var
  I,T: Integer;
{$IFDEF Shark64}
  D: TInt64;
  K: PInt64;
{$ELSE}
  R,L: LongWord;
  K: PIntArray;
{$ENDIF}
begin
  K := User;
{$IFDEF Shark64}
  Inc(K, 7);
  D := PInt64(Data)^;
  for I := 0 to 4 do
  begin
    D := D xor K^; Inc(K);
    D := TShark_Data(Shark_CD)[0, D shr 56 and $FF] xor
         TShark_Data(Shark_CD)[1, D shr 48 and $FF] xor
         TShark_Data(Shark_CD)[2, D shr 40 and $FF] xor
         TShark_Data(Shark_CD)[3, D shr 32 and $FF] xor

         TShark_Data(Shark_CD)[4, D shr 24 and $FF] xor
         TShark_Data(Shark_CD)[5, D shr 16 and $FF] xor
         TShark_Data(Shark_CD)[6, D shr  8 and $FF] xor
         TShark_Data(Shark_CD)[7, D        and $FF];
  end;
  D := D xor K^; Inc(K);
  D := (Int64(Shark_SD[D shr 56 and $FF]) shl 56) xor
       (Int64(Shark_SD[D shr 48 and $FF]) shl 48) xor
       (Int64(Shark_SD[D shr 40 and $FF]) shl 40) xor
       (Int64(Shark_SD[D shr 32 and $FF]) shl 32) xor
       (Int64(Shark_SD[D shr 24 and $FF]) shl 24) xor
       (Int64(Shark_SD[D shr 16 and $FF]) shl 16) xor
       (Int64(Shark_SD[D shr  8 and $FF]) shl  8) xor
       (Int64(Shark_SD[D        and $FF]));
  PInt64(Data)^ := D xor K^;
{$ELSE}
  Inc(PInteger(K), 14);
  L := PInt64(Data).L;
  R := PInt64(Data).R;
  for I := 0 to 4 do
  begin
    L := L xor K[0];
    R := R xor K[1];
```

```
    Inc(PInteger(K), 2);
    T := Shark_CD[0, R shr 23 and $1FE] xor
         Shark_CD[1, R shr 15 and $1FE] xor
         Shark_CD[2, R shr  7 and $1FE] xor
         Shark_CD[3, R shl  1 and $1FE] xor
         Shark_CD[4, L shr 23 and $1FE] xor
         Shark_CD[5, L shr 15 and $1FE] xor
         Shark_CD[6, L shr  7 and $1FE] xor
         Shark_CD[7, L shl  1 and $1FE];
    R := Shark_CD[0, R shr 23 and $1FE or 1] xor
         Shark_CD[1, R shr 15 and $1FE or 1] xor
         Shark_CD[2, R shr  7 and $1FE or 1] xor
         Shark_CD[3, R shl  1 and $1FE or 1] xor
         Shark_CD[4, L shr 23 and $1FE or 1] xor
         Shark_CD[5, L shr 15 and $1FE or 1] xor
         Shark_CD[6, L shr  7 and $1FE or 1] xor
         Shark_CD[7, L shl  1 and $1FE or 1];
    L := T;
  end;
  L := L xor K[0];
  R := R xor K[1];
  Inc(PInteger(K), 2);
  L := Integer(Shark_SD[L shr 24        ]) shl 24 xor
       Integer(Shark_SD[L shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[L shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[L        and $FF]);
  R := Integer(Shark_SD[R shr 24        ]) shl 24 xor
       Integer(Shark_SD[R shr 16 and $FF]) shl 16 xor
       Integer(Shark_SD[R shr  8 and $FF]) shl  8 xor
       Integer(Shark_SD[R        and $FF]);
  PInt64(Data).L := L xor K[0];
  PInt64(Data).R := R xor K[1];
{$ENDIF}
end;

procedure TCipher_Shark.Init(const Key; Size: Integer; IVector: Pointer);
var
  Log, ALog: array[0..255] of Byte;

  procedure InitLog;
  var
    I, J: Word;
  begin
    ALog[0] := 1;
    for I := 1 to 255 do
    begin
      J := ALog[I-1] shl 1;
      if J and $100 <> 0 then J := J xor $01F5;
      ALog[I] := J;
    end;
    for I := 1 to 254 do Log[ALog[I]] := I;
  end;

  function Transform(A: TInt64): TInt64;
  type
    TInt64Rec = packed record
                  Lo, Hi: Integer;
                end;

    function Mul(A, B: Integer): Byte;
    begin
      Result := ALog[(Log[A] + Log[B]) mod 255];
    end;

  var
    I,J: Byte;
    K,T: array[0..7] of Byte;
  begin
{$IFDEF Shark64}
```

```
    Move(TInt64Rec(A).Hi, K[0], 4);
    Move(TInt64Rec(A).Lo, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
{$ELSE}
    Move(A.R, K[0], 4);
    Move(A.L, K[4], 4);
    SwapIntegerBuffer(@K, @K, 2);
{$ENDIF}
    for I := 0 to 7 do
    begin
      T[I] := Mul(Shark_I[I, 0], K[0]);
      for J := 1 to 7 do T[I] := T[I] xor Mul(Shark_I[I, J], K[J]);
    end;
{$IFDEF Shark64}
    Result := T[0];
    for I := 1 to 7 do Result := Result shl 8 xor T[I];
{$ELSE}
    Result.L := T[0];
    Result.R := 0;
    for I := 1 to 7 do
    begin
      Result.R := Result.R shl 8 or Result.L shr 24;
      Result.L := Result.L shl 8 xor T[I];
    end;
{$ENDIF}
  end;

  function Shark(D: TInt64; K: PInt64): TInt64;
  var
    R,T: Integer;
  begin
{$IFDEF Shark64}
    for R := 0 to 4 do
    begin
      D := D xor K^; Inc(K);
      D := TShark_Data(Shark_CE)[0, D shr 56 and $FF] xor
           TShark_Data(Shark_CE)[1, D shr 48 and $FF] xor
           TShark_Data(Shark_CE)[2, D shr 40 and $FF] xor
           TShark_Data(Shark_CE)[3, D shr 32 and $FF] xor
           TShark_Data(Shark_CE)[4, D shr 24 and $FF] xor
           TShark_Data(Shark_CE)[5, D shr 16 and $FF] xor
           TShark_Data(Shark_CE)[6, D shr  8 and $FF] xor
           TShark_Data(Shark_CE)[7, D        and $FF];
    end;
    D := D xor K^; Inc(K);
    D := (Int64(Shark_SE[D shr 56 and $FF]) shl 56) xor
         (Int64(Shark_SE[D shr 48 and $FF]) shl 48) xor
         (Int64(Shark_SE[D shr 40 and $FF]) shl 40) xor
         (Int64(Shark_SE[D shr 32 and $FF]) shl 32) xor
         (Int64(Shark_SE[D shr 24 and $FF]) shl 24) xor
         (Int64(Shark_SE[D shr 16 and $FF]) shl 16) xor
         (Int64(Shark_SE[D shr  8 and $FF]) shl  8) xor
         (Int64(Shark_SE[D        and $FF]));
    Result := D xor K^;
{$ELSE}
    for R := 0 to 4 do
    begin
      D.L := D.L xor K.L;
      D.R := D.R xor K.R;
      Inc(K);
      T   := Shark_CE[0, D.R shr 23 and $1FE] xor
             Shark_CE[1, D.R shr 15 and $1FE] xor
             Shark_CE[2, D.R shr  7 and $1FE] xor
             Shark_CE[3, D.R shl  1 and $1FE] xor
             Shark_CE[4, D.L shr 23 and $1FE] xor
             Shark_CE[5, D.L shr 15 and $1FE] xor
             Shark_CE[6, D.L shr  7 and $1FE] xor
             Shark_CE[7, D.L shl  1 and $1FE];
```

```
      D.R := Shark_CE[0, D.R shr 23 and $1FE or 1] xor
             Shark_CE[1, D.R shr 15 and $1FE or 1] xor
             Shark_CE[2, D.R shr  7 and $1FE or 1] xor
             Shark_CE[3, D.R shl  1 and $1FE or 1] xor
             Shark_CE[4, D.L shr 23 and $1FE or 1] xor
             Shark_CE[5, D.L shr 15 and $1FE or 1] xor
             Shark_CE[6, D.L shr  7 and $1FE or 1] xor
             Shark_CE[7, D.L shl  1 and $1FE or 1];
      D.L := T;
    end;
    D.L := D.L xor K.L;
    D.R := D.R xor K.R;
    Inc(K);
    D.L := Integer(Shark_SE[D.L shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.L shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.L shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.L        and $FF]);
    D.R := Integer(Shark_SE[D.R shr 24 and $FF]) shl 24 xor
           Integer(Shark_SE[D.R shr 16 and $FF]) shl 16 xor
           Integer(Shark_SE[D.R shr  8 and $FF]) shl  8 xor
           Integer(Shark_SE[D.R        and $FF]);
    Result.L := D.L xor K.L;
    Result.R := D.R xor K.R;
{$ENDIF}
  end;

var
  T: array[0..6] of TInt64;
  A: array[0..6] of TInt64;
  K: array[0..15] of Byte;
  I,J,R: Byte;
  E,D: PInt64Array;
  L: TInt64;
begin
  InitBegin(Size);
  FillChar(K, SizeOf(K), 0);
  Move(Key, K, Size);
  InitLog;
  E := User;
  D := @E[7];
  Move(Shark_CE[0], T, SizeOf(T));
  T[6] := Transform(T[6]);
  I := 0;
{$IFDEF Shark64}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R] := K[I and $F];
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R] := A[R] shl 8 or K[I and $F];
    end;
  end;
  E[0] := A[0] xor Shark(0, @T);
  for R := 1 to 6 do E[R] := A[R] xor Shark(E[R - 1], @T);
{$ELSE}
  for R := 0 to 6 do
  begin
    Inc(I);
    A[R].L := K[I and $F];
    A[R].R := 0;
    for J := 1 to 7 do
    begin
      Inc(I);
      A[R].R := A[R].R shl 8 or A[R].L shr 24;
      A[R].L := A[R].L shl 8 or K[I and $F];
    end;
  end;
```

```
  L.L := 0;
  L.R := 0;
  L := Shark(L, @T);
  E[0].L := A[0].L xor L.L;
  E[0].R := A[0].R xor L.R;
  for R := 1 to 6 do
  begin
    L := Shark(E[R - 1], @T);
    E[R].L := A[R].L xor L.L;
    E[R].R := A[R].R xor L.R;
  end;
{$ENDIF}

  E[6] := Transform(E[6]);
  D[0] := E[6];
  D[6] := E[0];
  for R := 1 to 5 do D[R] := Transform(E[6-R]);

  FillChar(Log, SizeOf(Log), 0);
  FillChar(ALog, SizeOf(ALog), 0);
  FillChar(T, SizeOf(T), 0);
  FillChar(A, SizeOf(A), 0);
  FillChar(K, SizeOf(K), 0);
  InitEnd(IVector);
end;

class procedure TCipher_Square.GetContext(var ABufSize, AKeySize, AUserSize:
Integer);
begin
  ABufSize := 16;
  AKeySize := 16;
  AUserSize := 9 * 4 * 2 * SizeOf(LongWord);
end;

class function TCipher_Square.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    043h,09Ch,0A6h,0C4h,067h,0E8h,02Eh,047h
         DB    022h,095h,066h,085h,006h,039h,06Ah,0C9h
         DB    018h,021h,020h,0F7h,044h,036h,0F1h,061h
         DB    07Dh,014h,090h,0B1h,0A9h,068h,056h,0C7h
end;

procedure TCipher_Square.Encode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := User;
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);
  for I := 0 to 6 do
  begin
    AA := Square_TE[0, A        and $FF] xor
          Square_TE[1, B        and $FF] xor
          Square_TE[2, C        and $FF] xor
          Square_TE[3, D        and $FF] xor Key[0];
    BB := Square_TE[0, A shr  8 and $FF] xor
          Square_TE[1, B shr  8 and $FF] xor
          Square_TE[2, C shr  8 and $FF] xor
          Square_TE[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TE[0, A shr 16 and $FF] xor
          Square_TE[1, B shr 16 and $FF] xor
```

```
          Square_TE[2, C shr 16 and $FF] xor
          Square_TE[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TE[0, A shr 24        ] xor
          Square_TE[1, B shr 24        ] xor
          Square_TE[2, C shr 24        ] xor
          Square_TE[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);

    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SE[A        and $FF])        xor
                        LongWord(Square_SE[B        and $FF]) shl  8 xor
                        LongWord(Square_SE[C        and $FF]) shl 16 xor
                        LongWord(Square_SE[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SE[A shr  8 and $FF])        xor
                        LongWord(Square_SE[B shr  8 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SE[A shr 16 and $FF])        xor
                        LongWord(Square_SE[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SE[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SE[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SE[A shr 24       ])        xor
                        LongWord(Square_SE[B shr 24       ]) shl  8 xor
                        LongWord(Square_SE[C shr 24       ]) shl 16 xor
                        LongWord(Square_SE[D shr 24       ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Decode(Data: Pointer);
var
  Key: PIntArray;
  A,B,C,D: LongWord;
  AA,BB,CC: LongWord;
  I: Integer;
begin
  Key := @PIntArray(User)[9 * 4];
  A := PIntArray(Data)[0] xor Key[0];
  B := PIntArray(Data)[1] xor Key[1];
  C := PIntArray(Data)[2] xor Key[2];
  D := PIntArray(Data)[3] xor Key[3];
  Inc(PInteger(Key), 4);

  for I := 0 to 6 do
  begin
    AA := Square_TD[0, A        and $FF] xor
          Square_TD[1, B        and $FF] xor
          Square_TD[2, C        and $FF] xor
          Square_TD[3, D        and $FF] xor Key[0];
    BB := Square_TD[0, A shr  8 and $FF] xor
          Square_TD[1, B shr  8 and $FF] xor
          Square_TD[2, C shr  8 and $FF] xor
          Square_TD[3, D shr  8 and $FF] xor Key[1];
    CC := Square_TD[0, A shr 16 and $FF] xor
          Square_TD[1, B shr 16 and $FF] xor
          Square_TD[2, C shr 16 and $FF] xor
          Square_TD[3, D shr 16 and $FF] xor Key[2];
    D  := Square_TD[0, A shr 24        ] xor
          Square_TD[1, B shr 24        ] xor
          Square_TD[2, C shr 24        ] xor
          Square_TD[3, D shr 24        ] xor Key[3];

    Inc(PInteger(Key), 4);
    A := AA; B := BB; C := CC;
  end;

  PIntArray(Data)[0] := LongWord(Square_SD[A        and $FF])        xor
                        LongWord(Square_SD[B        and $FF]) shl  8 xor
```

```
                        LongWord(Square_SD[C        and $FF]) shl 16 xor
                        LongWord(Square_SD[D        and $FF]) shl 24 xor Key[0];
  PIntArray(Data)[1] := LongWord(Square_SD[A shr  8 and $FF])        xor
                        LongWord(Square_SD[B shr  8 and $FF]) shl  8 xor
                        LongWord(Square_SD[C shr  8 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr  8 and $FF]) shl 24 xor Key[1];
  PIntArray(Data)[2] := LongWord(Square_SD[A shr 16 and $FF])        xor
                        LongWord(Square_SD[B shr 16 and $FF]) shl  8 xor
                        LongWord(Square_SD[C shr 16 and $FF]) shl 16 xor
                        LongWord(Square_SD[D shr 16 and $FF]) shl 24 xor Key[2];
  PIntArray(Data)[3] := LongWord(Square_SD[A shr 24        ])        xor
                        LongWord(Square_SD[B shr 24        ]) shl  8 xor
                        LongWord(Square_SD[C shr 24        ]) shl 16 xor
                        LongWord(Square_SD[D shr 24        ]) shl 24 xor Key[3];
end;

procedure TCipher_Square.Init(const Key; Size: Integer; IVector: Pointer);
type
  PSquare_Key = ^TSquare_Key;
  TSquare_Key = array[0..8, 0..3] of LongWord;
var
  E,D: PSquare_Key;
  T,I: Integer;
begin
  InitBegin(Size);
  E := User;
  D := User; Inc(D);
  Move(Key, E^, Size);
  for T := 1 to 8 do
  begin
    E[T, 0] := E[T -1, 0] xor ROR(E[T -1, 3], 8) xor 1 shl (T - 1); D[8 -T, 0]
:= E[T, 0];
    E[T, 1] := E[T -1, 1] xor E[T, 0];                              D[8 -T, 1]
:= E[T, 1];
    E[T, 2] := E[T -1, 2] xor E[T, 1];                              D[8 -T, 2]
:= E[T, 2];
    E[T, 3] := E[T -1, 3] xor E[T, 2];                              D[8 -T, 3]
:= E[T, 3];
    for I := 0 to 3 do
      E[T -1, I] :=     Square_PHI[E[T -1, I]        and $FF]      xor
                    ROL(Square_PHI[E[T -1, I] shr  8 and $FF],  8) xor
                    ROL(Square_PHI[E[T -1, I] shr 16 and $FF], 16) xor
                    ROL(Square_PHI[E[T -1, I] shr 24        ], 24);
  end;
  D[8] := E[0];
  InitEnd(IVector);
end;

{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386


procedure FindVirtualMethodAndChange(AClass: TClass; MethodAddr, NewAddress:
Pointer);
type
  PPointer = ^Pointer;
const
  PageSize = SizeOf(Pointer);
var
  Table: PPointer;
  SaveFlag: DWORD;
begin
  Table := PPointer(AClass);
  while Table^ <> MethodAddr do Inc(Table);
  if VirtualProtect(Table, PageSize, PAGE_EXECUTE_READWRITE, @SaveFlag) then
  try
    Table^ := NewAddress;
  finally
    VirtualProtect(Table, PageSize, SaveFlag, @SaveFlag);
```

```
  end;
end;
  {$ENDIF}
{$ENDIF}

{$IFDEF VER_D3H}
procedure ModuleUnload(Module: Integer);
var
  I: Integer;
begin
  if IsObject(FCipherList, TStringList) then
    for I := FCipherList.Count-1 downto 0 do
      if FindClassHInstance(TClass(FCipherList.Objects[I])) = Module then
        FCipherList.Delete(I);
end;
{$ENDIF}

initialization
{$IFDEF UseASM}
  {$IFNDEF 486GE}  // не підтримується для <= CPU 386
  if CPUType <= 3 then  // CPU <= 386
  begin
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Encode,
                                                 @TCipher_Blowfish.Encode386);
    FindVirtualMethodAndChange(TCipher_Blowfish, @TCipher_Blowfish.Decode,
                                                 @TCipher_Blowfish.Decode386);
  end;
  {$ENDIF}
{$ENDIF}
{$IFDEF VER_D3H}
  AddModuleUnloadProc(ModuleUnload);
{$ENDIF}
{$IFNDEF ManualRegisterClasses}
  RegisterCipher(TCipher_3Way, '', '');
  RegisterCipher(TCipher_Blowfish, '', '');
  RegisterCipher(TCipher_Gost, '', '');
  RegisterCipher(TCipher_IDEA, '', 'не комерційний');
  RegisterCipher(TCipher_Q128, '', '');
  RegisterCipher(TCipher_SAFER_K40, 'SAFER-K40', '');
  RegisterCipher(TCipher_SAFER_SK40, 'SAFER-SK40', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K64, 'SAFER-K64', '');
  RegisterCipher(TCipher_SAFER_SK64, 'SAFER-SK64', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K128, 'SAFER-K128', '');
  RegisterCipher(TCipher_SAFER_SK128, 'SAFER-SK128', 'Keyscheduling');
  RegisterCipher(TCipher_SCOP, '', '');
  RegisterCipher(TCipher_Shark, '', '');
  RegisterCipher(TCipher_Square, '', '');
  RegisterCipher(TCipher_TEA, 'TEA', '');
  RegisterCipher(TCipher_TEAN, 'TEA розширений', '');
  RegisterCipher(TCipher_Twofish, '', '');
{$ENDIF}
finalization
{$IFDEF VER_D3H}
  RemoveModuleUnloadProc(ModuleUnload);
{$ENDIF}
  FCipherList.Free;
  FCipherList := nil;
end.
```

Файл main.pas основної програми

```
unit Main;

interface

{$I VER.INC}

// Підключення бібліотек

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

// Опис змінних

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

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

```
var
  MainForm: TMainForm;

implementation

// Початок опису реалізацій функцій нейронного аналізу

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

{$R *.DFM}

// визначення розміру файлу, який дається на аналіз

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

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

// Вибір файлу

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

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

// обробка нажаття клавіші

procedure TMainForm.MItemExitClick(Sender: TObject);
begin
  Close;
end;
```

```
//Створення головної форми

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

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

procedure TMainForm.FormActivate(Sender: TObject);
begin

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

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

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

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

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

procedure TMainForm.BtnCalcHashClick(Sender: TObject);
var
  FileSize: Integer;
//  Hash: THash;
//  HashClass: THashClass;
//  Digest: String;
//  Stream: TFileStream;
//  Buf: array[0..7] of Integer;
//  Len: Integer;
begin
  EKeyChange(nil);
  EBase16.Text := '';
  EBase64.Text := '';
```

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

    HashManager.CalcFile(EHashFile.Text);

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

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

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

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

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

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


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

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

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

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


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

procedure TMainForm.CBCipherModeClick(Sender: TObject);
const
  sMode : array[TCipherMode] of String =
```

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

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

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

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

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

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

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

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

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

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

```
    CipherManager.DecodeFile(FENCFile, FDECFile);

      LDecodeTime.Caption := Counter(FileSize);

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

// Підпрограмма тестування файла навчання

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

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

// Підпрограма обробки швидкості хешування
```

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

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

// Підпрограма обробки швидкості шифрування

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

// Визначаємо фрагмент якого коду обробляється

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

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

procedure TMainForm.MItemCipherVectorClick(Sender: TObject);
{ генеруємо TestVector для шифру та вставляємо Codefragment to Clipboard }
var
  Data, Caption: String;
begin
  with CipherManager.CipherClass.Create('', nil) do
  try
    Data := ClassName;
    Mode := cmCTS;
    Init(PChar(Data)^, Length(Data), nil);
    SetLength(Data, 32);
```

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

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

// Процедури обробки натискання клавіш

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

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

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

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

GenForm.pas – Побудова форм та основних обробників клавіш

```
unit GenForm;

interface

// Підключення бібліотек

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

// Опис головного об'єкту

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

var
  GForm: TGForm;

implementation

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

//Початок роботи підпрограми

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

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

// Обробник закриття форм

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

  finally
    Stream.Free;
  end;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

end;

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

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

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

end;

procedure TGForm.MACwithRFC2104Click(Sender: TObject);

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

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

  HUser := DefaultHashClass;

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

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

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

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

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

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

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

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

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

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

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

  finally
    MAC.Release;
  end;

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

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

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

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

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

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

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

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

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

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

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

  finally
    MAC.Release;
  end;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  finally
    MAC.Release;
    Stream.Free;
  end;

end;

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

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

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

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

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

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

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

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

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

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

  DoInfo('будуємо TAN list для "Маслюков Олександр Сергійович"', clBlue);

  TANList := CreateTANList(SeedTANList, SeedTAN, 'Маслюков Олександр
Сергійович', 54, LastTAN);

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

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

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

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

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

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

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

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

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

  finally
    TANList.Free;
  end;
end;

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

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

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


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

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

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

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

    Protect; //

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

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

    S := DecodeString(S);

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

    Protect;

//-----------------------------------------------------------------------------
    DoInfo('3. Шифрування/дешифрування буфера, "' +StrToFormat(@Buffer,
Sizeof(Buffer), Format) + '"', clBlue);
    InitKey('DEC Part I', nil);

    EncodeBuffer(Buffer, Buffer, SizeOf(Buffer));
    DoInfo('Шифрування:'#9+ StrToFormat(@Buffer, Sizeof(Buffer), Format),
clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
    Done;

    DecodeBuffer(Buffer, Buffer, SizeOf(Buffer));

    DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(@Buffer, Sizeof(Buffer),
Format), clWindowText);
    DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    Protect;

//-----------------------------------------------------------------------------
    DoInfo('4. Шифрування/дешифрування потоку, "Partial Stream En/Decryption"',
clBlue);
```

```
    InitKey('DEC Part I', nil);

    Stream := TMemoryStream.Create;
    try
      S := 'Partial Stream En/Decryption';
      Stream.Write(PChar(S)^, Length(S));

      Stream.Position := 8;
      EncodeStream(Stream, Stream, 6);

      DoInfo('Шифрування:'#9+ StrToFormat(Stream.Memory, Stream.Size, Format),
clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);
      Done;

      Stream.Position := 8;
      DecodeStream(Stream, Stream, 6);

      DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(Stream.Memory,
Stream.Size, Format), clWindowText);
      DoInfo('MAC'#9+CalcMAC(Format), clWindowText);

    finally
      Stream.Free;
    end;

//---------------------------------------------------------------------------
    DoInfo('5. Різні режими шифрування', clBlue);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      InitKey('DEC', nil);
      S := EncodeString('Тест нейронною мережею зашифрованого рядка');
      DoInfo('Шифрування:'#9+ StrToFormat(PChar(S), Length(S), Format),
clWindowText);

      Done;

      S := DecodeString(S);
      DoInfo('Взлам нейронною мережею:'#9+ StrToFormat(PChar(S), Length(S),
Format), clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('6. Використовувати TProtection-Method CodeString() with any
Protection', clBlue);

  with CUser.Create('', nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('Без шифрування', clBlue);

    InitKey('DEC', nil);
    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
```

```
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено  TRandom_LFSR("DEC")', clBlue);
    Protection := TRandom_LFSR.Create('DEC', 400, False, nil);
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Захищено THash_MD5(TMAC_RFC2104("DEC"))', clBlue);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('DEC', nil));
    InitKey('DEC', nil);

    for K := cmCTS to cmECB do
    begin
      DoInfo('Mode: ' + sMode[K], clGreen);
      Mode := K;

      S := CodeString('Тест нейронною мережею зашифрованого рядка', paEncode,
Format);
      DoInfo('Шифрування:'#9+ S, clWindowText);

      S := CodeString(S, paDecode, Format);
      DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);
    end;

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('7. Використовуємо TProtections Methods', clBlue);

  with CUser.Create('', nil) do
  try
    HashClass := THash_MD5;  // установлюємо інші HashClass for the InitKey()
    InitKey('DEC', nil);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paEncode/paDecode)', clGreen);

    S := CodeString('CodeString()', paEncode, Format);
    DoInfo('Шифрування:'#9+ S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Взлам нейронною мережею:'#9+ S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paScramble)', clGreen);

    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paScramble, Format);
    DoInfo('Scramble:'#9+ S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('CodeString(paWipe)', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
```

```
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);
    S := CodeString('CodeString()', paWipe, Format);
    DoInfo('Взломано:'#9+ S, clWindowText);

  finally
    Free;
  end;
//-------------------------------------------------------------------------
  DoInfo('8. A secure зашифрований ', clBlue);
//  demonstrate a Multi-En/Decryption that використовує 3 Ciphers in a Chain.

  with TCipher_Blowfish.Create('DEC',
         TCipher_IDEA.Create('DEC',
           TCipher_GOST.Create('DEC',
             TRandom_LFSR.Create('Scramble', 128, False, nil)))) do
  try
    S := CodeString('Добрий DEC Part I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);
  finally
    Free;
  end;

end;

procedure TGForm.CipherMACClick(Sender: TObject);
const
  sMode: array[TCipherMode] of String = ('CTS-MAC', 'CBC-MAC', 'CFB-MAC',
                                         'invalid', 'invalid',
                                         'CTS-MAC', 'CBC-MAC', 'CFB-MAC');
var
  CUser: TCipherClass;
  K: TCipherMode;
  I: Integer;
begin
  M.Clear;
  CUser := TCipher_Blowfish; // вибираємо шифр для взламу

  DoInfo('Посилається автентифікаційний код з шифром', clRed);
  DoInfo('Користувач визначив шифр як : ' + GetCipherName(CUser), clMaroon);
  DoInfo('Відбувається самотестування в нейронній мережі Ciphers', clBlue);
  DoInfo('CUser:'#9 + sSelfTest[CUser.SelfTest], clWindowText);
  DoInfo('SCOP:'#9 + sSelfTest[TCipher_SCOP.SelfTest], clWindowText);

//-------------------------------------------------------------------------
  DoInfo('1. MAC для ParamStr(0), DEMO.EXE', clBlue);
  with CUser.Create('DEC', nil) do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() in MAC Mode: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() in MAC Mode: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
```

```
  end;

//---------------------------------------------------------------------------
  DoInfo('2. MAC для ParamStr(0), DEMO.EXE Захищено Blowfish("Secret")',
clBlue);
  with CUser.Create('DEC', TCipher_Blowfish.Create('Secret', nil)) do
  try
    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      DoInfo('EncodeFile() in MAC Mode: ' + sMode[K], clGreen);
      EncodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);

      DoInfo('DecodeFile() in MAC Mode: ' + sMode[K], clGreen);
      DecodeFile(ParamStr(0), '');
      for I := 1 to 3 do
        DoInfo(IntToStr(I) + ': ' + sMode[K] + #9 + CalcMAC(Format),
clWindowText);
    end;
  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('3. MAC''s with TProtection Method CodeString', clBlue);

  with CUser.Create('DEC', nil) do
  try
// визначіть  HMAC-MD5-LFSR128-SCOP protection :-)
//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP', clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

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

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for MD5',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('SecRet 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

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

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for LFSR',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('SecRet 2', 128, False,
                       TCipher_SCOP.Create('Secret 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
```

```
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;

//---------------------------------------------------------------------------
    DoInfo('Protection HMAC-MD5-LFSR128-SCOP with other Password for SCOP',
clGreen);
    Protection := THash_MD5.Create(TMAC_RFC2104.Create('Secret 1',
                    TRandom_LFSR.Create('Secret 2', 128, False,
                       TCipher_SCOP.Create('SecRet 3', nil))));

    for K := cmCTSMAC to cmCFBMAC do
    begin
      Mode := K;
      CodeString('Добрий DEC Part I', paCalc, fmtNONE);
      DoInfo(sMode[K] + #9 + CalcMAC(Format), clWindowText);
    end;
  finally
    Free;
  end;
 CodeBuffer(), CodeString()
//   CodeStream(), CodeFile()
// - with Action = paCalc it's the Result from CodeString()
end;

procedure TGForm.UsingfromRandoms1Click(Sender: TObject);
var
  I: Integer;
  S,SaveState: String;
  Buf: array[0..15] of Byte;
begin
  M.Clear;

  DoInfo('Random using', clRed);
//---------------------------------------------------------------------------
  DoInfo('1. TRandom_LFSR Instance with Period 2^400-1', clBlue);

  with TRandom_LFSR.Create('', 400, False, nil) do
  try
//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC Part I"', clBlue);
    Seed('DEC Part I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, Seed "DEC ParT I"', clBlue);
    Seed('DEC ParT I', 10);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, default Seed', clBlue);
    Seed('', 0);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з 1000, randomized Seed', clBlue);
    Seed('', -1);
```

```
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1000 to 1000, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1000) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('Change Period to 2^2032-1', clGreen);
    Size := 2032;

//---------------------------------------------------------------------------
    DoInfo('20 випадкових чисел з -1 to 1, randomized Seed', clBlue);
    Seed('', -1);
    S := '';
    for I := 1 to 20 do
      S := S + IntToStr( Int(-1) ) + ',';
    SetLength(S, Length(S) -1);
    DoInfo(S, clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed', clBlue);
    Seed('', 0);

    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
    DoInfo('randomized Buffer, default Seed with saveing of State', clBlue);
    DoInfo('Switch back to Period 2^128-1', clGreen);
    Size := 128;
    Protection := TCipher_Blowfish.Create('DEC', nil);
    Seed('', -1);          SaveState := State;
    DoInfo('SaveState:' + InsertBlocks(SaveState, #9, #10, 64), clGreen);
// випадковий Buffer
    Buffer(Buf, SizeOf(Buf));

    DoInfo(StrToFormat(@Buf, SizeOf(Buf), Format), clWindowText);

    Seed('', -1); //
    DoInfo('Стан відновлено з SaveState', clGreen);

    State := SaveState;
    SetLength(S, Sizeof(Buf));
    Buffer(PChar(S)^, Length(S));

    DoInfo(StrToFormat(PChar(S), Length(S), Format), clWindowText);

  finally
    Free;
  end;

//---------------------------------------------------------------------------
  DoInfo('2. Глобальна випадкова змінна "RND"', clBlue);
// RND is per default TRandom_LFSR('', 128);
  RND.Seed('', 0);
  RND.Buffer(Buf, SizeOf(Buf));
  DoInfo(StrToFormat(@Buf, Sizeof(Buf), Format), clWindowText);

//---------------------------------------------------------------------------
```

```
  DoInfo('3. TProtection методи з TRandom_LFSR', clBlue);
  with TRandom_LFSR.Create('DEC', 128, False, nil) do
  try
//-------------------------------------------------------------------------------
    DoInfo('Кодування / декодування нейронною мережею з TRandom', clGreen);

    S := CodeString('Добрий DEC Part I', paEncode, Format);
    DoInfo('Encrypted:'#9+S, clWindowText);
    S := CodeString(S, paDecode, Format);
    DoInfo('Decrypted:'#9+S, clWindowText);

//-------------------------------------------------------------------------------
    DoInfo('Утаємничення з TRandom', clGreen);

    S := CodeString('Добрий DEC Part I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

    DoInfo('BasicSeed changed from: '+ SysUtils.Format('$%0.8x to $%0.8x',
[BasicSeed, BasicSeed +1]), clGreen);
    BasicSeed := BasicSeed +1; // використовує інший BasicSeed
    S := CodeString('Добрий DEC Part I', paScramble, Format);
    DoInfo('Scrambled:'#9+S, clWindowText);

//-------------------------------------------------------------------------------
    DoInfo('Взломано з TRandom', clGreen);
// paWipe is normaly used with CodeBuffer(), CodeFile() and CodeStream()
    S := CodeString('Добрий DEC Part I', paWipe, Format);
    DoInfo('Wiped:'#9+S, clWindowText);

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

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

end;

end.
```