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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20ПЗ
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Кудрик К.А.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *125 "*Кібербезпека*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Кібербезпека*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

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

Кудрику Кирилу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації*

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

затверджені наказом вищого навчального закладу № 138-02 від 01.04.2024 року

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки комплексного криптографічного захисту інформації*

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

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

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

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

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Кудрик К.А.
(прізвище та ініціали)

АНОТАЦІЯ

Кудрик К.А. Програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки комплексного криптографічного захисту інформації.

Метою розробки є програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації.

Результат роботи – програмна реалізація системи кібербезпеки комплексного криптографічного захисту інформації.

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

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

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

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: кібербезпека, криптографічний захист інформації

ABSTRACT

Kudryk K.A. Software of the cyber security system of complex cryptographic protection of information. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of complex cryptographic protection of information.

The goal of the development is the software of the cyber security system of complex cryptographic protection of information.

The result of the work is the software implementation of the cyber security system of complex cryptographic protection of information.

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

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

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 Sydney environment.

Keywords: cyber security, cryptographic protection of information

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГПВЧ	–	генератор псевдовипадкових чисел
ЗКЗІ	–	засоби криптографічного захисту інформації
CRC	–	алгоритм підрахунку контрольних сум
PBA	–	Pre-boot Authentication

ВСТУП

Актуальність теми. У цей час стає усе більше затребуваним використання одного засобу захисту інформації (ЗЗІ) для рішення цілого ряду завдань, спрямованих на захист системи. До таких завдань відносяться:

– захист від несанкціонованого доступу;

– розмежування доступу користувачів і процесів до інформації;

– криптографічний захист файлів за допомогою механізмів шифрування й електронного цифрового підпису.

Багато власників компаній хочуть вирішити всі ці завдання, не бажаючи використовувати для цього кілька незалежних пристроїв, кожний з яких реалізує одну або декілька з необхідних функцій по захисту інформації. Таким чином, необхідно реалізовувати комплексний захист інформації. При цьому, як правило, уважається, що чим більше функцій захисту інформації реалізовано в одній системі, тим краще. А ще краще, якщо всі ці функції реалізовані в одному пристрої.

Власники багатьох компаній не хочуть використовувати пристрої, що одночасно реалізують всі перераховані функції захисту інформації в силу того, що частина функцій у їхніх системах уже реалізована за допомогою інших пристроїв, або ж, у силу особливостей функціонування їхньої системи, у цих функціях взагалі немає необхідності. У цьому випадку компанії воліють купувати ЗЗІ, що реалізують тільки необхідні їм функції захисту інформації й не переплачувати за наявність інших.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації.

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

– Огляд існуючих систем комплексного криптографічного захисту інформації.

– Дослідження системи кібербезпеки комплексного криптографічного захисту інформації.

– Програмна реалізація системи кібербезпеки комплексного криптографічного захисту інформації.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі комплексного криптографічного захисту інформації.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

На даному етапі розвитку інформаційних технологій відбувається поступове витиснення паперового документообігу електронним, що актуально, практично, для організацій будь-яких областей діяльності. Електронний документообіг має незаперечні переваги, але є й недоліки, зокрема, необхідність захисту переданих електронних документів. Тому системи електронного документообігу звичайно містять у собі різні підсистеми захисту інформації.

Найбільше гостро питання захисту документообігу стоїть для організацій, що мають територіально-розподілену структуру. Такі організації можуть мати кілька локальних обчислювальних мереж, розташованих у різних місцях, у тому числі в різних регіонах Україні, і змушені використовувати для передачі інформації різні глобальні обчислювальні мережі загального користування, наприклад, мережа Internet. Використання Internet для передачі даних найбільше небезпечно через величезне число підключених до Internet користувачів. Будьте впевнені, що серед них обов'язково найдуться ті, кому виняткова цікава передана Вами або для Вас інформація, і дуже сумнівно, що ця інформація буде використана ними Вам у благо.

Погрози з боку таких користувачів можна розділити на дві основні категорії: погроза конфіденційності інформації, тобто, несанкціоноване ознайомлення з документами, і погроза цілісності інформації, тобто, можливі спроби змінити в процесі передачі електронний документ або нав'язати помилковий документ під видом легально прийшов. Відповідно, існують два основних методи захисту від подібних погроз: шифрування для забезпечення конфіденційності електронних документів і електронний підпис для забезпечення цілісності й точного встановлення авторства документа.

1.2 Область застосування

Областю застосування є комплексна система криптографічного захисту інформації.

Шифрування

Шифрування – це процес перетворення відкритих даних у закриті по певному криптографічному алгоритму з використанням секретного ключового елемента – ключа шифрування. Стандартом шифрування в Україні є алгоритм ДСТУ 28147:2009. Даний алгоритм є симетричним, тобто для зашифрування й наступного розшифрування інформації використовується те саме криптографічне перетворення.

Секретний елемент криптографічного перетворення – ключ шифрування – може зберігатися, наприклад, у файлі на дискеті, або на якому-небудь іншому ключовому носії (смарт-карті, USB-ключі й т.п.). Необхідно, щоб всі користувачі, що припускають обмінюватися зашифрованими документами, одержали якийсь набір ключів шифрування, що дозволило б адресатам розшифровувати документи, попередньо зашифровані відправниками.

Найпростіший випадок – всі абоненти комп'ютерної мережі організації одержують той самий секретний ключ шифрування. Як і всі простої, така схема має ряд недоліків:

– Всі абоненти мережі мають той самий ключ шифрування. Таким чином, будь-які зашифровані цим ключем документи можуть бути розшифровані будь-яким абонентом мережі, тобто, неможливо відправити якийсь документ якому-небудь абонентові особисто.

– При компрометації ключа шифрування (втраті, розкраданні й т.д.) під погрозою порушення конфіденційності виявиться весь документообіг, ключі шифрування прийде терміново міняти. Якщо ж, наприклад, факт компрометації ключа шифрування виявлений не відразу, залишиться тільки догадуватися, скільки документів (і якої важливості) встиг прочитати зловмисник.

– Такі ключі необхідно передавати «з рук у руки», тобто неможливо, наприклад, переслати їх по електронній пошті, що незручно.

Перші два недоліки усуваються за допомогою ключової системи типу «повна матриця». «Повна матриця» містить матрицю ключів для зв'язку «кожний з кожним» (ключі парного зв'язку), тобто матриця містить ключі для зв'язку абонентів попарно. Це означає, що кожний із ключів матриці доступний тільки двом з абонентів мережі. Кожний абонент забезпечується «мережним набором» – рядком ключів з даної матриці, призначених для його зв'язку з іншими абонентами мережі. Таким чином, існує можливість посилати документи кому-небудь, зашифрувавши їх на ключі «для двох», що робить документ недоступним для інших. Простіше й з компрометацією – при втраті якого-небудь ключа варто боятися лише за ті документи, які посилали власниками конкретного ключа один одному. Відповідно, новий ключ замість скомпрометованого необхідно замінити тільки у двох абонентів, а не в усіх.

При необхідності послати той самий документ декільком абонентам у зашифрованому виді, його варто зашифровувати кілька разів (по числу адресатів), а потім ще й не переплутати, кому який із зашифрованих файлів відсилати.

Залишається необхідність передачі ключів «з рук у руки».

Проблема передачі ключів вирішується шляхом застосування схеми відкритого розподілу ключів. Це означає, що за допомогою певного алгоритму ключ шифрування «ділиться» на секретну й відкриту частини. Секретна частина, називана «секретним ключем», зберігається в його власника, а відкрита частина («відкритий ключ») передається всім іншим абонентам мережі. Таким чином, кожний абонент мережі має у своєму розпорядженні свій власний секретний ключ і набір відкритих ключів всіх інших абонентів. За допомогою свого секретного ключа й відкритого ключа абонента-адресата абонент-відправник обчислює ключ парного зв'язку, за допомогою якого зашифровує документи, призначені даному одержувачеві. Одержувач же, за допомогою свого секретного

ключа й наявного в нього відкритого ключа відправника, обчислює той же ключ парного зв'язку, за допомогою якого може ці документи розшифрувати.

Таким чином, шляхом використання схеми з відкритим розподілом ключів досягаються ті ж позитивні моменти, що й при використанні схеми «повна матриця», але також і нейтралізується недолік – проблема розподілу ключів. Відкриті ключі можна розподіляти вільно по відкритих каналах зв'язку, оскільки, навіть маючи повний набір відкритих ключів всіх абонентів мережі, зловмисник не зможе розшифрувати конфіденційний документ, оскільки він призначений конкретному адресатові. Те ж саме ставиться й до інших абонентів мережі, оскільки документ зашифрований на ключі парного зв'язку, що можуть обчислити тільки відправник і одержувач – в інших попросту не вистачає вихідних даних для обчислення ключа парного зв'язку.

Ще одне істотне достоїнство відкритого розподілу ключів полягає в тому, що ті самі ключі можуть бути використані й для шифрування документів, і для електронного підпису. При використанні інших ключових схем це недосяжно.

Електронний підпис

Електронний цифровий підпис (ЕЦП) – засіб, що дозволяє на основі криптографічних методів установити авторство й цілісність електронного документа. В Україні також існує стандарт ЕЦП – алгоритм ДСТУ 4145-2002.

Ключова система даного алгоритму повністю ідентична описаній вище схемі шифрування з відкритим розподілом ключів. Кожний користувач має власний секретний ключ, за допомогою якого він підписує електронні документи таким чином, що, не маючи секретного ключа, підробити його підпис неможливо. Парний секретному відкритий ключ використовується для перевірки ЕЦП. Результатом перевірки ЕЦП є встановлення автора документа й твердження, вірний чи ні підпис конкретного документа, тобто чи дійсно вона поставлена автором документа й чи не був документ змінений у процесі його передачі.

Комплексний захист інформації

Природно, для захисту й конфіденційності, і цілісності інформації варто використовувати в комплексі шифрування й ЕЦП, що також можна сполучити з яким-небудь додатковим сервісом, наприклад, стиском інформації (архівацією). Як приклад можна привести алгоритм створення спеціалізованого архіву.

Створюваний у такий спосіб файл-архів можна передавати по мережі без яких-небудь побоювань. При створенні архіву вихідні файли підписуються на секретному ключі користувача (за алгоритмом ДСТУ 4145-2002), після чого файли стискуються й одержуваний у результаті стиску архів шифрується на випадковому тимчасовому ключі (за алгоритмом ДСТУ 28147:2009). Абоненти, яким призначається архів, можуть розшифрувати його за допомогою записаного в архів зашифрованого тимчасового ключа. Тимчасовий ключ зашифровується на парно-зв'язному ключі, що обчислюється із секретного ключа відправника й відкритого ключа абонента-адресата. Таким чином, досягаються наступні цілі:

– Передані електронні документи забезпечуються електронним підписом, що захищає їх від порушення цілісності або підміни.

– Документи передаються в захищеному виді, що забезпечує їхня конфіденційність.

– Користувачі-адресати можуть розшифрувати документи, використовуючи свій секретний ключ і відкритий ключ відправника.

– Користувачі, яким не призначається даний архів, не можуть прочитати його вміст, оскільки не мають тимчасового ключа й не можуть його обчислити.

– Додатковий сервіс – зменшення розміру, обумовлене архівацією.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Програмний комплекс еталонних тестових засобів (ПК "ЕТАЛОН")

ПК "Еталон" призначений для формування періодичних послідовностей тестових кодових комбінацій.

ПК "Еталон" використовується при проведенні досліджень побічних електромагнітних випромінювань і наведень засобів обчислювальної техніки (ЗОТ).

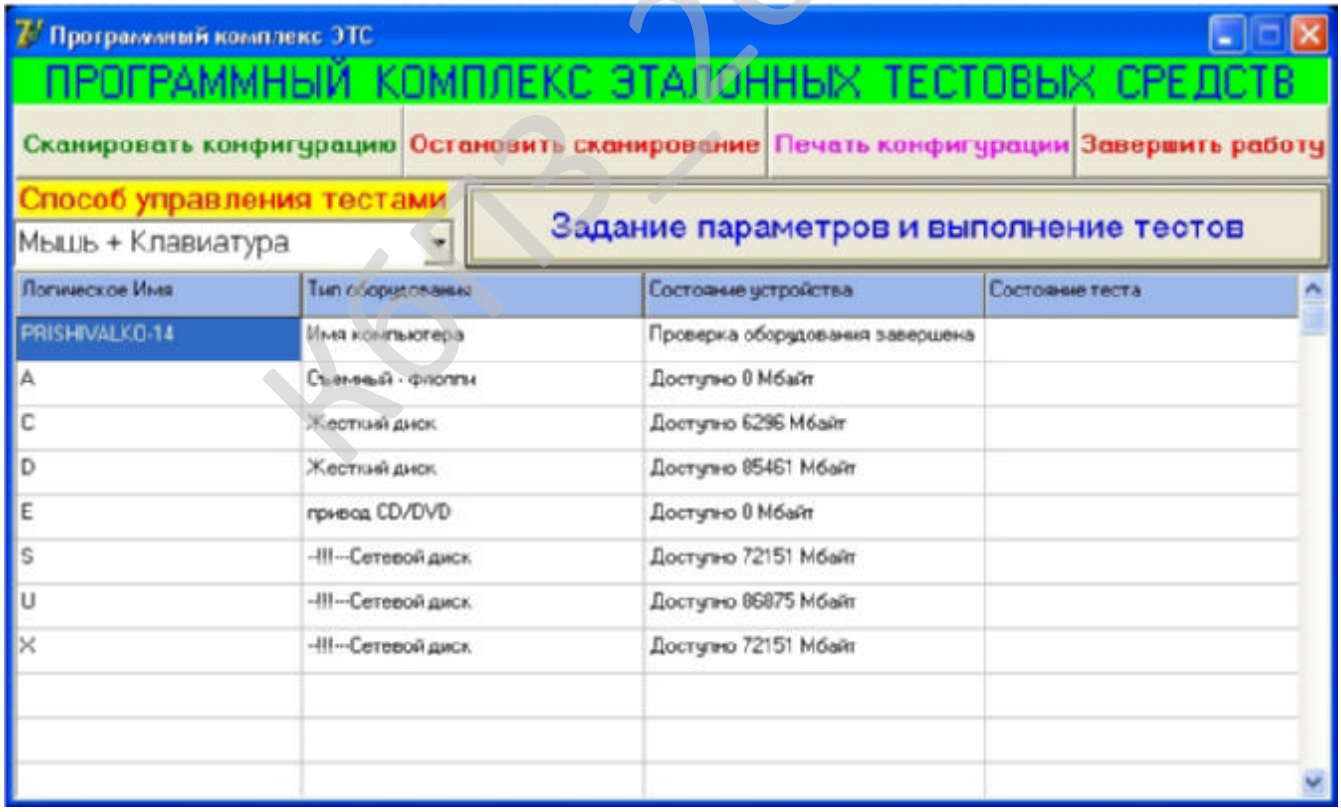

Рисунок 2.1 – Інтерфейс користувача ПК "Еталон"

Технічні характеристики:

1. ПК "Еталон" забезпечує завдання спеціальних тестових режимів функціонування наступних пристроїв ЗОТ при проведенні їхніх спеціальних досліджень:

– оперативної пам'яті ПЕОМ;

– флеш-пам'яті;

– накопичувачів на жорстких магнітних дисках (НЖМД);

– накопичувачів на дисках CD-R; CD-RW, DVD;

– накопичувачів на гнучких магнітних дисках (НГМД);

– відеомонітора;

– матричного принтера;

– струминного принтера;

– лазерного принтера;

– мережних адаптерів;

– клавіатури.

2. ПК "Еталон" має можливість завдання будь-якого набору двійкового коду, використовуваного для обміну з досліджуваним пристроєм.

3. ПК "Еталон" забезпечує роботу в інтерактивному режимі по зручному графічному користувальницькому інтерфейсі із системою меню й забезпеченням індикації на екрані відеомонітора тестуємої ПЕОМ ознаки роботи тестової програми. ПК "Еталон" надає можливість операторові управляти ходом виконання тестових програм (задавати конкретні значення байтів переданої інформації, задавати логічне ім'я накопичувача, що перевіряється, зупиняти й повторно запускати тестову програму).

Програмний комплекс "КриптоТестер"

Програмний комплекс "Криптотестер" призначений для автоматизації процесів тестування при проведенні експертизи або сертифікаційних випробувань програмних реалізацій стандартних криптографічних алгоритмів.

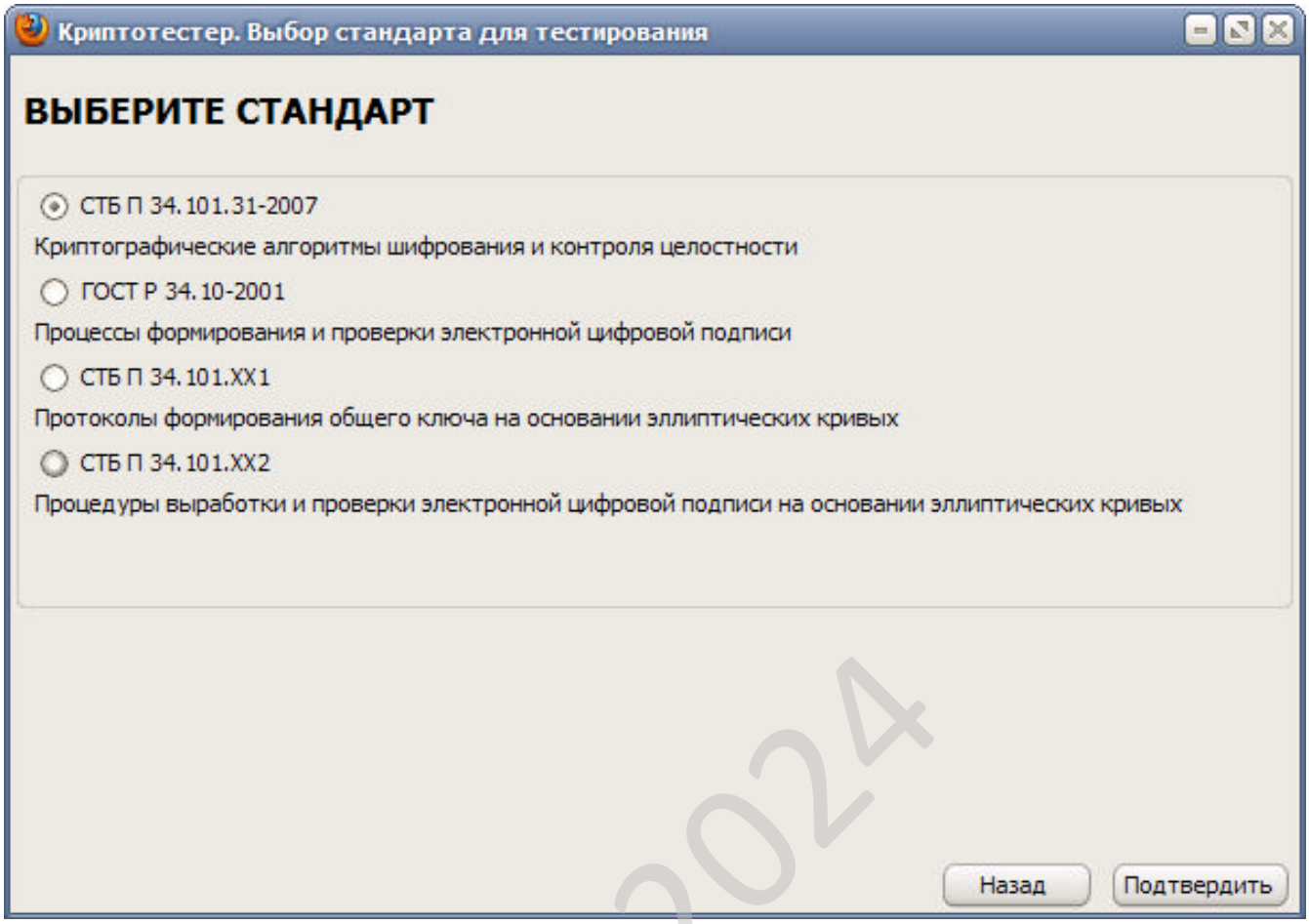

Рисунок 2.2 – Інтерфейс користувача програмного комплексу "КриптоТестер"

Дозволяє:

– вибрати стандарт криптографічного перетворення із заданого набору;

– вибрати методи тестування (метод тестів, побудованих на еталонний вхідний і вихідний даних, і метод тестів, побудованих на випадково сформованих вхідні й не еталонних вихідних даних);

– вибрати спосіб уведення даних для тестування (уведення вхідних даних з файлу, вручну й шляхом автоматичного генерування вхідних даних);

– визначити обсяг тестових даних;

– провести повторне тестування на використовуваних даних;

– провести перевірку параметрів еліптичній кривій відповідно до обраного стандарту;

– зареєструвати вихідні дані й результати тестування в базі даних;

– переглянути результати тестування у формі.

Переваги застосування:

– власна розробка без не декларованих можливостей;

– одержання об'єктивної оцінки якості реалізації стандартного криптографічного алгоритму на основі розрахункових результатів;

– можливість багаторазового виконання навантажувального тестування;

– одержання високого ступеня надійності результатів випробувань;

– значне зменшення тимчасових і трудових витрат при проведенні сертифікаційних випробувань засобів криптографічного захисту інформації.

Захищена інформаційно-аналітична система

Захищена інформаційно-аналітична система (ЗІАС) призначена для забезпечення інформаційної підтримки, електронного обміну даними й документами, а також для надання персоналу державних органів інформації й аналітичних матеріалів, необхідних для прийняття рішень на різних рівнях державного керування.

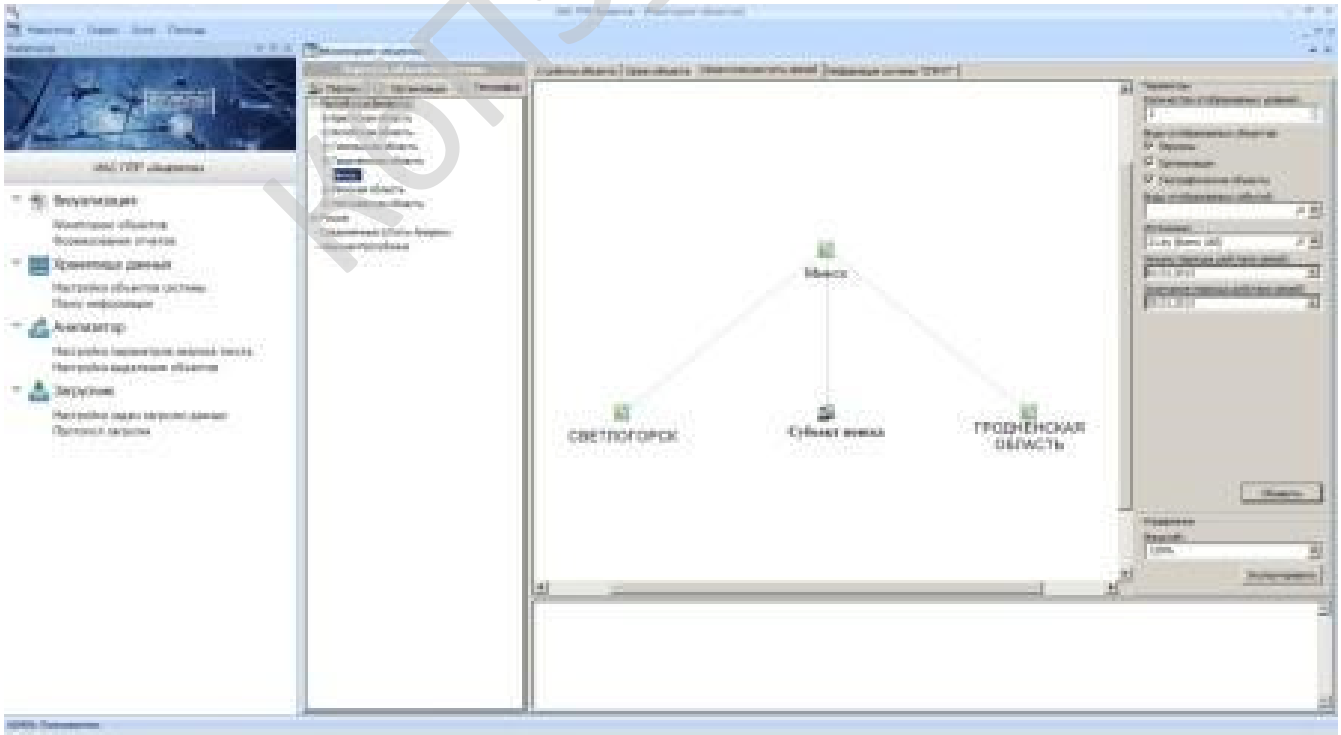

Рисунок 2.3 – Інтерфейс користувача ЗІАС

Функції системи

ЗІАС забезпечує:

– настроювання функціональних можливостей системи шляхом редагування метаданих і алгоритмів;

– збір і завантаження даних;

– розмежування доступу по ролях (уведення інформації, побудова звітів, аналіз інформації, адміністрування системи) і контроль їхніх обігів;

– консолідацію інформації, що складає в об'єднанні інформації, отриманої з декількох джерел і зв'язку, що має, очищенню, верифікації й індентифікації інформації;

– оповіщення адміністратора про помилки завантаження й зміні формату зовнішнього джерела, а також про всі інциденти, пов'язаних зі спробами порушень політики безпеки;

– ведення структурованого архіву виділених об'єктів;

– забезпечення рівня захисту збереженої й оброблюваної інформації, певного нормативними правовии й технічними нормативними правовими актами, і відповідного рівня конфіденційності (таємності) цієї інформації й захисту від фальсифікації збереженої інформації;

– можливість побудови багаторівневих семантичних мереж на основі наявних даних про відносини й зв'язки об'єктів, що цікавлять;

– редагування семантичних схем;

– експорт звітних документів у зовнішні формати даних;

– ведення захищеного журналу роботи системи;

– захист інформації в сховище;

– резервування даних і відновлення системи після збоїв;

– формування документів системи й звітів в електронному виді із застосуванням електронного цифрового підпису;

– взаємодія з іншими інформаційними системами.

Програмне забезпечення гарантованого знищення інформації

Програмне забезпечення гарантованого знищення інформації (ПЗГЗІ) дозволяє проводити гарантоване знищення інформації конфіденційного характеру (за винятком відомостей, що становлять державні секрети), що зберігається на накопичувачах на жорстких магнітних дисках (НЖМД), флеш-накопичувачах і інших пристроях, обумовлених операційною системою (ОС) Windows як "HARDDISK".

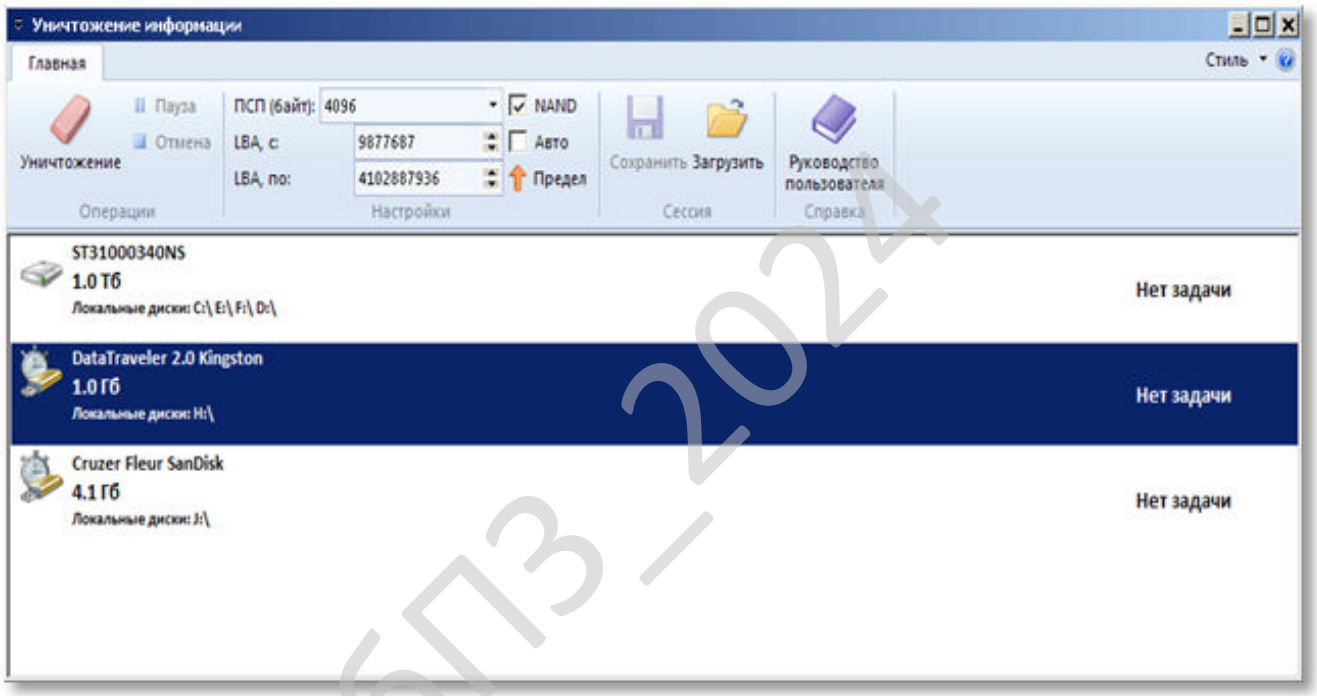

Рисунок 2.4 – Інтерфейс користувача ПЗГЗІ

Знищення інформації виконується програмним способом методом багаторазового перезапису даних.

При перезаписі даних використовується наступний алгоритм:

1. Для НЖМД:

– цикл 1 – запис нулів;

– цикл 2 – запис одиниць;

– цикл 3-6 – запис нулів, що чергуються, і одиниць;

– цикл 7 – запис псевдовипадкової послідовності.

2. Для накопичувачів з NAND-Пам'яттю:

– цикл 1 – запис нулів;

– цикл 2 – запис одиниць;

– цикл 3 – запис псевдовипадкової послідовності.

Припустимий діапазон знищення даних обмежений значенням логічної лінійної адреси (Logical block address, LBA). Інформація на ушкоджені або недоступні в результаті апаратного захисту носіях інформації не знищується.

Знищення інформації програмним способом дозволяє повторно використовувати носії інформації.

Комплекс програмних засобів організації захищеного каналу передачі даних "ZCHANNEL"

Комплекс програмних засобів організації захищеного каналу передачі даних (комплекс) призначений для забезпечення криптографічного захисту переданої інформації на базі національних стандартів, використовує двосторонню періодичну автентифікацію сторін і можливість конфігурування дозволених шляхів з'єднання передачі даних усередині каналу, що захищається.

Комплекс є крос-платформним і функціонує під ОС Windows, Linux.

Носій ключової інформації забезпечує надійне зберігання ключової інформації й виконання криптографічних перетворень.

Комплекс створює захищений канал між клієнтом і сервером, забезпечує захист з'єднання будь-якого прикладного протоколу, що працює через TCP-з'єднання без динамічного відкриття портів, зокрема HTTP, SMTP, POP3, IMAP, FTP (passive mode), NFS, SQL і т.д.

Комплекс відповідає наступним стандартам в області захисту інформації:

– міжнародним стандартам і рекомендаціям X.509, PKI, PKCS#7, PKCS#11, S/MIME, CMS, а також СТБ 34.101.19, СТБ 34.101.23;

– СТБ 1176.2 і СТБ П 34.101.45 с використанням СТБ 34.101.47, СТБ 34.101.31 для вироблення й перевірки ЕЦП;

– СТБ 34.101.31 для обчислення значення геш-функції;

– ДСТУ 28147:2009, СТБ 34.101.31 для шифрування даних.

Комплекс дозволяє створювати довірений захищений канал між двома й більше вузлами в мережах передачі даних для виключення можливості погрози прослуховування трафіку, без необхідності вносити зміни в роботу прикладного програмного забезпечення клієнта й сервера.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

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

Підвищення якості й швидкодії інструментів

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

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

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки комплексного криптографічного захисту інформації.

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

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

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У наших українських умовах легко уявити собі ситуацію, коли з комп'ютера, що зберігає конфіденційну інформацію, витягається жорсткий диск і підключається до іншого комп'ютера. А там бажаючий ознайомитися з інформацією знає свій пароль і має права адміністратора. З урахуванням такої можливості покладатися на один тільки пароль досить легковажно.

Разом з тим шифрування безсило проти різних програмних і апаратних закладок, «троянів», мережного злому й інших атак, яким може піддатися працюючий комп'ютер із завантаженими ключами шифрування, коли користувач або адміністратор може просто не знати, що на комп'ютер проникнув сторонній. У цьому випадку зловмисник тим або іншим способом прикидається легальним користувачем, і одержує доступ до інформації також, як і легальний користувач. На жаль, шифрування не вміє перевіряти права доступу користувачів на доступ до інформації.

Тому, шифрування даних – це лише один з важливих елементів системи інформаційної безпеки, але зовсім не достатній. Необхідна наявність грамотна настроєної системи розмежування доступу, контролю цілісності операційного середовища, засобів виявлення проникнень, антивірусного й антитроянського захисту й т.д.

У різних системах можуть використовуватися різні способи шифрування даних. Це може бути шифрування на рівні файлів, або шифрування на рівні секторів диска.

Аналізуючи дані з відомих джерел, можна рекомендувати використовувати файл-контейнер для захисту даних окремих користувачів на їхніх комп'ютерах; у цьому випадку навантаження не занадто високе, і падіння

продуктивності не так помітно. Більше проста процедура установки й керування такою системою навіть якоюсь мірою компенсує ці недоліки.

Для захисту ж корпоративних серверів, до яких пред'являються більше високі вимоги по продуктивності й надійності, рекомендується використовувати блокове шифрування розділів диска. У цьому випадку додаткові, до того ж, як правило, разові роботи з інсталяції й настроювання системи захисти цілком виправдуються більше високим ступенем надійності й меншим падінням продуктивності.

Програмне забезпечення призначене для комплексної системи криптографічного захисту інформації . Дія програми побудована на принципі шифрування файлів на ключі, параметри якого залежать від пароля, що вводиться користувачем.

Вона забезпечує:

– безпечне зберігання конфіденційних матеріалів на жорстких дисках комп’ютерів;

– розмежування доступу до інформації, яка зберігається на комп’ютері, який використовується декількома особами;

– захист інформації, яка переміщується на переносних та кишенькових комп’ютерах, інших засобах зберігання інформації;

– пересилку конфіденційних матеріалів на любих носіях інформації з використанням електронної пошти або інших систем обміну інформацією.

Програма має високу стійкість до злому, і розкриття захищених файлів при переборі всіх можливих комбінацій ключа на сучасних ЕОМ вимагає десятків років. Програма також підтримує систему відновлення пароля шифрування.

Комплексна система криптографічного захисту інформації призначена для реалізації в прикладному програмному забезпеченні функцій криптографічного захисту інформації – застосування геш-функції й шифрування.

Основне призначення системи – забезпечення застосування геш-функції й шифрування, у тому числі й у юридично значимому електронному документообігу.

Система може застосовуватися практично в будь-яких системах різного рівня й призначення, що потребують у використанні засобів геш-функції й шифрування. У їхнє число входять:

– Системи електронного документообігу і юридично значимого електронного документообігу.

– Системи транспортного призначення (електронна пошта, системи гарантованої доставки електронних повідомлень).

– Системи забезпечення прийняття управлінських рішень, інтеграційні середовища.

– Системи автоматизації бізнес-процесів.

– Кадрові системи.

– Бухгалтерські системи, у тому числі 1С.

– WEB-додатки й інформаційні портали.

– Архівні системи й електронні сховища документів.

– Інформаційно-довідкові системи.

– Прикладні системи персонального використання.

Система комплексного криптографічного захисту інформації може використовуватися й безпосередньо користувачем MS Windows для формування геш-функції й шифрування файлів у провіднику MS Windows.

Система комплексного криптографічного захисту інформації призначена для зручності застосування геш-функції всіма, як просто користувачами, так і програмістами будь-якої кваліфікації. Для користувачів вона дає можливість роботи з геш-функцією файлів безпосередньо із провідника Windows. Програмісти можуть швидко й просто вмонтувати роботу з геш-функціями в нові або модифікувати вже експлуатовані програми.

Головною перевагою системи комплексного криптографічного захисту інформації є те, що вона бере на себе складних і потребуючих певних знань і кваліфікації функції спілкування із засобами комплексного криптографічного захисту інформації інформації (ЗКЗІ) і сховищами сертифікатів.

Для звичайного користувача система надає візуальний інтерфейс із набором команд, а для програміста COM інтерфейс із набором високорівневих операцій роботи з геш-функцією й сертифікатами. При цьому як користувачеві, так і програмістові досить мати загальні поняття про організацію застосування геш-функції й не лізти в нетрі криптографії. Щоб організувати застосування геш-функції й шифрування з використанням системи комплексного криптографічного захисту інформації досить вивчити представлений в даному дипломному проекті.

Система комплексного криптографічного захисту інформації не вимагає специфічних знань по криптографії не тільки від користувача, але й від програміста або системного адміністратора. Всі складності в роботі із криптографічними механізмами бере на себе сама. Це стає можливим завдяки тому, що в системі комплексного криптографічного захисту інформації використовується високорівневий інтерфейс і для звичайного користувача, і для користувача – програміста. Система дозволяє швидко й надійно доповнити будь-яку прикладну систему засобами геш-функції й шифрування.

Достоїнством системи комплексного криптографічного захисту інформації є й те, що це – тиражуємий продукт українського розроблювача. Це означає облік особливостей українських криптопровайдерів і низьку вартість володіння.

Для використання системи комплексного криптографічного захисту інформації бажано ознайомитися з азами організації застосування геш-функції й шифрування.

Після установки системи на комп'ютер розроблювача він відразу ж зможе використовувати всі її можливості для програмування роботи з геш-функцією й шифруванням у середовищі Windows.

Розроблене програмне забезпечення дозволяє виконувати наступні функції захисту інформації:

– проводити шифрування;

– будувати геш-функції масивів інформації;

– реалізовувати алгоритми генерації псевдовипадкових чисел;

– підраховувати контрольну суму файлів.

Розглянемо ці технології захисту інформації більш докладно.

Алгоритми шифрування реалізовані у програмному забезпеченні базуються на симетричних алгоритмах.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема системи. Програмне забезпечення структурно складається з наступних блоків:

– Головний модуль програми.

– Шифрування файлів та папок.

– Шифрування даних на змінних носіях інформації.

– Шифрування даних, які передаються по мережі.

– Шифрування листів e-mail.

– Створення зашифрованих архівів, що саморозпаковуються.

– Генератор псевдовипадкових чисел.

– Перевірка цілісності та автентичності файлів.

– Створення віртуального зашифрованого диску.

– Створення паролю.

– База даних алгоритмів шифрування.

– База даних геш-функцій.

– База даних алгоритмів генерації псевдовипадкових чисел (ГПВЧ).

– База даних алгоритмів підрахунку контрольних сум (CRC).

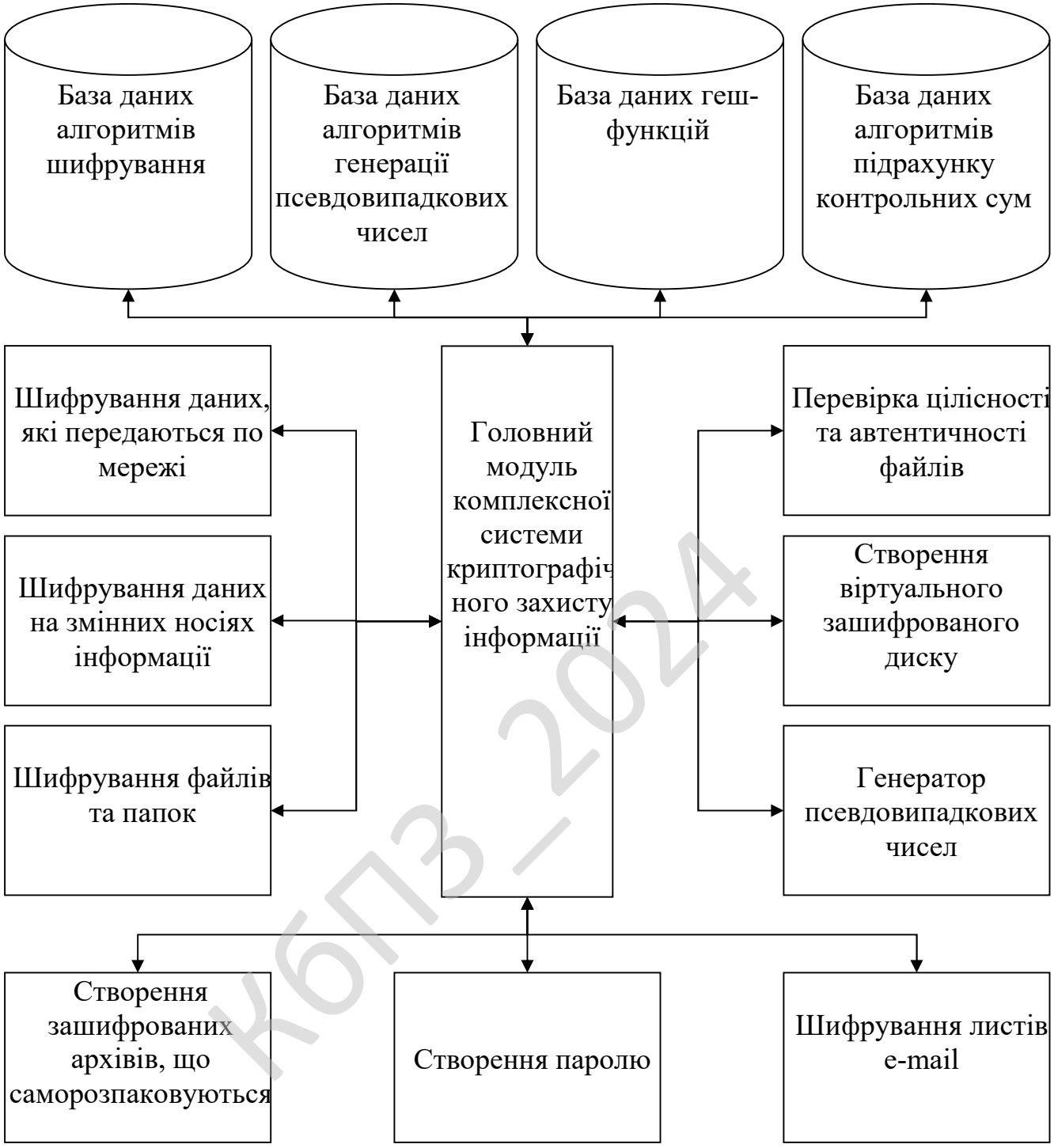

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Алгоритм DES

Найпоширенішим і найбільш відомим алгоритмом симетричного шифрування є DES (Data Encryption Standard). Алгоритм був розроблений в 1977 році, в 1980 році був прийнятий NIST (National Institute of Standards and Technolody США) як стандарт (FIPS PUB 46).

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти з у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

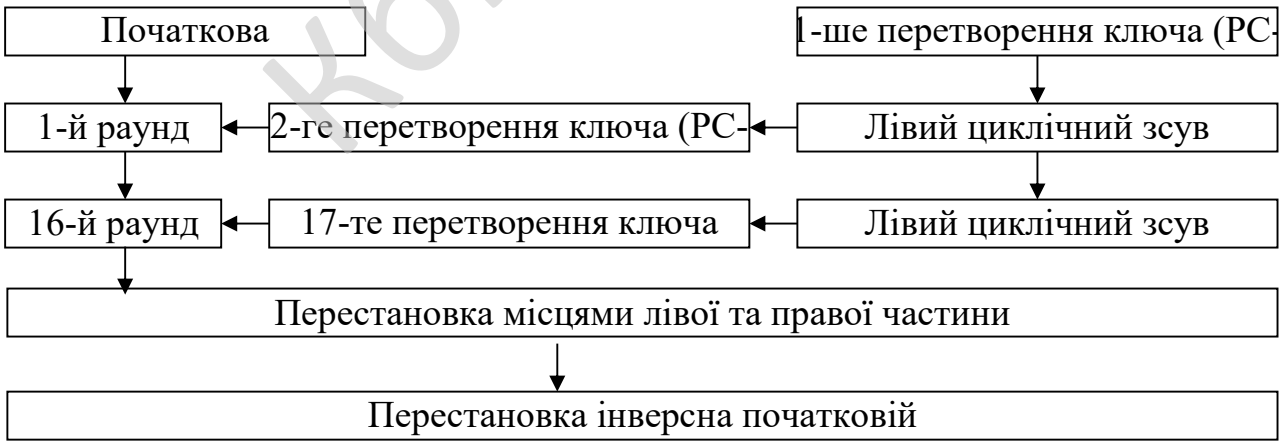

Рисунок 3.2 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

Потрійний DES

У цей час основним недоліком DES вважається маленька довжина ключа, тому вже давно почали розроблятися різні альтернативи цьому алгоритму шифрування. Один з підходів полягає в тому, щоб розробити новий алгоритм, і успішний тому приклад – IDEA. Інший підхід припускає повторне застосування шифрування за допомогою DES з використанням декількох ключів.

Алгоритм Blowfish

Blowfish є мережею Фейштеля, у якої кількість ітерацій дорівнює 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину в межах 448 біт. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко.

Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автентифікація сторін і узгодження загальних параметрів і секретів. Класичним прикладом подібних додатків є мережна взаємодія. При реалізації на 32-бітних мікропроцесорах з великим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа й шифрування даних.

Розширення ключа перетворить ключ довжиною, принаймні, 448 біт у кілька масивів підключей загальною довжиною 4168 байт.

В основі алгоритму лежить мережа Фейштеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа й даних. Операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключей. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних.

Алгоритм IDEA

IDEA (International Data Encryption Algorithm) є блоковим симетричним алгоритмом шифрування, розробленим Сюдзя Гавкіт (Xuejia Lai) і Джеймсом Массей (James Massey) зі швейцарського федерального інституту технологій. Первісна версія була опублікована в 1990 році. Переглянута версія алгоритму, посилена засобами захисту від диференціальних криптографічних атак, була представлена в 1991 році й докладно описана в 1992 році.

IDEA є одним з декількох симетричних криптографічних алгоритмів, якими спочатку передбачалося замінити DES.

Принципи розробки

IDEA є блоковим алгоритмом, що використовує 128-бітовий ключ для шифрування даних блоками по 64 біта.

Метою розробки IDEA було створення щодо стійкого криптографічного алгоритму з досить простою реалізацією.

Криптографічна стійкість

Наступні характеристики IDEA характеризують його криптографічну стійкість:

1. Довжина блоку: довжина блоку повинна бути достатньою, щоб сховати всі статистичні характеристики вихідного повідомлення. З іншого боку, складність реалізації криптографічної функції зростає експоненціально відповідно до розміру блоку. Використання блоку розміром в 64 біта в 90-і роки означало достатню силу. Більше того, використання режиму шифрування CBC говорить про подальше посилення цього аспекту алгоритму.

2. Довжина ключа: довжина ключа повинна бути досить великою для того, щоб запобігти можливості простого перебору ключа. При довжині ключа 128 біт IDEA вважається досить безпечним.

3. Конфузія: зашифрований текст повинен залежати від ключа складним і заплутаним способом.

4. Дифузія: кожний біт незашифрованого тексту повинен впливати на кожний біт зашифрованого тексту. Поширення одного незашифрованого біта на велику кількість зашифрованих біт приховує статистичну структуру незашифрованого тексту. Визначити, як статистичні характеристики зашифрованого тексту залежать від статистичних характеристик незашифрованого тексту, повинне бути непросто. IDEA із цього погляду є дуже ефективним алгоритмом.

В IDEA два останніх пункти виконуються за допомогою трьох операцій. Це відрізняє його від DES, де все побудовано на використанні операції XOR і маленьких нелінійних S-boxes.

Генератори випадкових чисел

Випадкові числа відіграють важливу роль при використанні криптографії в різних мережних додатках, що відносяться до безпеки. Зробимо короткий огляд вимог, пропонованих до випадкових чисел у додатках мережної безпеки, а потім розглянемо кілька способів створення випадкових чисел.

Вимоги до випадкових чисел

Більшість алгоритмів мережної безпеки, заснованих на криптографії, використовують випадкові числа. Двома основними вимогами до послідовності випадкових чисел є випадковість і непередбачуваність.

Випадковість

Звичайно при створенні послідовності псевдовипадкових чисел передбачається, що дана послідовність чисел повинна бути випадкової в деякому певному статистичному змісті. Наступні два критерії використовуються для доказу того, що послідовність чисел є випадковою:

1. Однорідний розподіл: розподіл чисел у послідовності повинне бути однорідним; це означає, що частота появи кожного числа повинна бути приблизно однаковою.

2. Незалежність: жодне значення в послідовності не повинне залежати від інших.

Хоча існують тести, що показують, що послідовність чисел відповідає деякому розподілу, такому як однорідний розподіл, тесту для "доказу" незалежності немає. Проте, можна підібрати набір тестів для доказу того, що послідовність є залежною. Загальна стратегія припускає застосування набору таких тестів доти, поки не буде впевненості, що незалежність існує.

Непередбачуваність

У додатках, таких як взаємна автентифікація й генерація ключа сесії, немає твердої вимоги, щоб послідовність чисел була статистично випадкової, але члени послідовності повинні бути непередбачені. При "правильній" випадковій послідовності кожне число статистично не залежить від інших чисел і, отже, непередбачено. Однак правильні випадкові числа на практиці використовуються досить рідко, частіше послідовність чисел, що повинна бути випадкової, створюється деяким алгоритмом. У цьому випадку необхідно, щоб супротивник не міг угадати наступні елементи послідовності, ґрунтуючись на знанні попередніх елементів і використовуваного алгоритму.

Джерела випадкових чисел

Джерела дійсно випадкових чисел знайти важко. Фізичні генератори шумів, такі як детектори подій іонізуючої радіації, газові розрядні трубки й конденсатор, що тече, можуть бути такими джерелами. Однак ці пристрої в додатках мережної безпеки застосовуються обмежено. Проблеми також викликають грубі атаки на такі пристрої. Альтернативним рішенням є створення набору з великої кількості випадкових чисел і опублікування його в деякій книзі. Проте, і такі набори забезпечують дуже обмежене джерело чисел у порівнянні з тією кількістю, що потрібна додаткам мережної безпеки. Більше того, хоча набори із цих книг дійсно забезпечує статистичну випадковість, вони передбачувані, тому що супротивник може одержати їхню копію.

Таким чином, додатки, що шифрують, використовують для створення випадкових чисел спеціальні алгоритми. Ці алгоритми детерміновані й, отже, створюють послідовність чисел, що не є статистично випадковою. Проте, якщо алгоритм гарний, отримана послідовність буде проходити багато тестів на випадковість. Такі числа часто називають **псевдовипадковими числами**.

Розглянемо кілька алгоритмів генерації випадкових чисел.

Генератори псевдовипадкових чисел

Першою широко використовуваною технологією створення випадкового числа був алгоритм, запропонований Лехмером, що відомий як метод лінійного конгруента. Цей алгоритм параметризується чотирма числами в такий спосіб:

m	Модуль (підстава системи)	$m > 0$
a	Множник	$0 \leq a < m$
c	Збільшення	$0 \leq c < m$
X_0	Початкове значення або зерно (seed)	$0 \leq X_0 < m$

Послідовність випадкових чисел $\{X_n\}$ виходить за допомогою наступної ітераційної рівності:

$$X_{n+1} = (a\, X_n + c) \bmod m.$$

Якщо m, a й c є цілими, то створюється послідовність цілих чисел у діапазоні $0 \leq X_n < m$.

Вибір значень для a, c и m є критичним для розробки гарного генератора випадкових чисел.

Очевидно, що m повинне бути дуже більшим, щоб була можливість створити багато випадкових чисел. Вважається, що m повинне бути приблизно дорівнючим максимальному позитивному цілому числу для даного комп'ютера. Таким чином, звичайно m близько або дорівнює 2^{31}.

Існує три критерії, які використовуються при виборі генератора випадкових чисел:

1. Функція повинна створювати повний період, тобто повинні існувати всі числа між 0 і m до того, як створювані числа почнуть повторюватися.

2. Створювана послідовність повинна з'являтися випадково. Послідовність не є випадкової, тому що вона створюється детерміновано, але різні статистичні тести, які можуть застосовуватися, повинні показувати, що послідовність випадкова.

3. Функція повинна ефективно реалізовуватися на 32-бітних процесорах, а в перспективі на 64-бітних.

Значення a, c и m повинні бути обрані таким чином, щоб ці три критерії виконувалися. Відповідно до першого критерію можна показати, що якщо m є простим і c = 0, то при певному значенні a період, створюваний функцією, буде дорівнює m-1. Для 32-бітної арифметики відповідне просте значення $m = 2^{31} - 1$. Таким чином, функція створення псевдовипадкових чисел має вигляд:

$$X_{n+1} = (a\ X_n) \bmod (2^{31} - 1).$$

Тільки невелике число значень a задовольняє всім трьом критеріям. Одне з таких значень є $a = 7^5 = 16807$, що використовувалося в сімействі комп'ютерів IBM 360. Цей генератор широко застосовується й пройшов більше тисячі тестів, більше, ніж всі інші генератори псевдовипадкових чисел.

Сила алгоритму лінійного конгруента в тім, що якщо співмножник і модуль (підстава) відповідним чином підібрані, то результуюча послідовність чисел буде статистично невідзначна від послідовності, що є випадкової з набору 1, 2, ..., m-1. Але не може бути випадковості в послідовності, отриманої з використанням алгоритму, незалежно від вибору початкового значення X_0. Якщо значення обране, то числа, що залишилися, у послідовності будуть визначені. Це завжди враховується при криптоаналізі.

Якщо супротивник знає, що використовується алгоритм лінійного конгруента, і якщо відомі його параметри ($a = 7^5$, $c = 0$, $m = 2^{31} - 1$), то, якщо розкрито одне число, вся послідовність чисел стає відома. Навіть якщо супротивник знає тільки, що використовується алгоритм лінійного конгруента, знання невеликої частини послідовності досить для визначення параметрів

алгоритму й всіх наступних чисел. Припустимо, що супротивник може визначити значення X_0, X_1, X_2, X_3. Тоді :

$$X_1 = (a\ X_0 + c) \bmod m.$$

$$X_2 = (a\ X_1 + c) \bmod m.$$

$$X_3 = (a\ X_2 + c) \bmod m.$$

Ці рівності дозволяють знайти a, c и m.

Таким чином, хоча алгоритм і є гарним генератором псевдовипадкової послідовності чисел, бажано, щоб реально використовувана послідовність була непередбаченою, оскільки в цьому випадку знання частини послідовності не дозволить визначити майбутні її елементи. Ця мета може бути досягнута декількома способами. Наприклад, використання внутрішніх системних годинників для модифікації потоку випадкових чисел. Один зі способів застосування годинників складається в перезапуску послідовності після N чисел, використовуючи поточне значення годин по модулі m у якості нового початкового значення. Інший спосіб складається в простому додаванні значення поточного часу до кожного випадкового числа по модулю m.

Криптографічно створені випадкові числа

У криптографічних додатках доцільно шифрувати випадкові числа, що виходять. Найчастіше використовується три способи.

Циклічне шифрування

У цьому випадку застосовується спосіб створення ключа сесії з майстра-ключа. Лічильник з періодом N використовується як вхід у пристрій, що шифрує. Наприклад, у випадку використання 56-бітного ключа DES може застосовуватися лічильник з періодом 2^{56}. Після кожного створеного ключа значення лічильника збільшується на 1. Таким чином, псевдовипадкова послідовність, отримана за даною схемою, має повний період: кожне вихідне значення X_0, X_1,...X_{N-1} засновано на різних значеннях лічильника й, отже, $X_0 \neq X_1 \neq X_{N-1}$. Так як майстер-ключ захищений, легко показати, що будь-який секретний ключ не залежить від знання одного або більше попередніх секретних ключів.

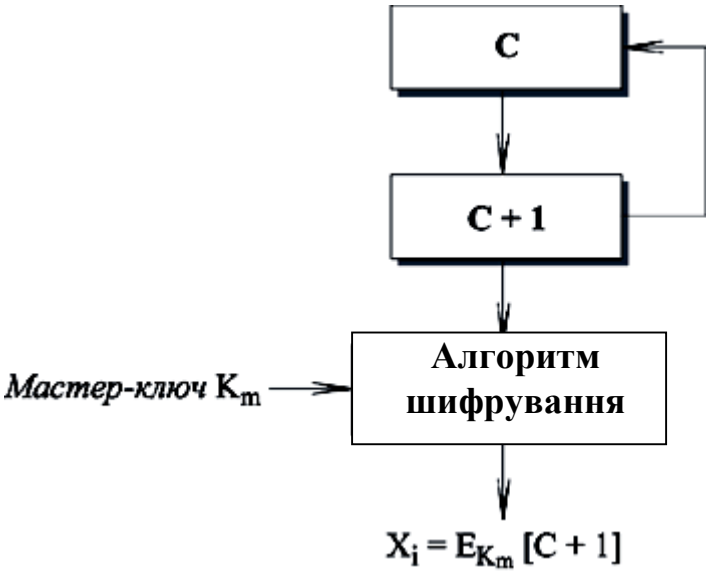

Рисунок 3.3 – Циклічне шифрування

Для подальшого посилення алгоритму вхід повинен бути виходом повноперіодичного генератора псевдовипадкових чисел, а не простою послідовністю.

Режим Output Feedback DES

Режим OFB DES може застосовуватися для генерації ключа, аналогічно тому, як він використовується для потокового шифрування. Помітимо, що виходом кожної стадії шифрування є 64-бітне значення, з якого тільки ліві j бітів подаються назад для шифрування. 64-бітні виходи становлять послідовність псевдовипадкових чисел з гарними статистичними властивостями.

Геш-функції

Вимоги до геш-функцій

Геш-функцією називається однобічна функція, призначена для одержання дайджесту або "відбитків пальців" файлу, повідомлення або деякого блоку даних.

Геш-код створюється функцією H:

$$h = H(M),$$

де M є повідомленням довільної довжини й h є геш-кодом фіксованої довжини.

Розглянемо вимоги, яким повинна відповідати геш-функція для того, щоб вона могла використовуватися в якості автентифікатора повідомлення. Розглянемо дуже простий приклад геш-функції. Потім проаналізуємо кілька підходів до побудови геш-функції.

Геш-функція H, що використовується для автентифікації повідомлень, повинна мати наступні властивості:

1. Геш-функція H повинна застосовуватися до блоку даних будь-якої довжини.
2. Геш-функція H створює вихід фіксованої довжини.
3. H (M) відносно легко (за поліноміальний час) обчислюється для будь-якого значення M.
4. Для будь-якого даного значення геш-коду h розрахунково неможливо знайти M таке, що H (M) = h.
5. Для будь-якого даного x розрахунково неможливо знайти y ≠x, що H (y) = H (x).
6. Розрахунково неможливо знайти довільну пару (x, y) таку, що H(y) = H(x).

Перші три властивості вимагають, щоб геш-функція створювала геш-код для будь-якого повідомлення.

Четверта властивість визначає вимогу однобічності геш-функції: легко створити геш-код по даному повідомленню, але неможливо відновити повідомлення по даному геш-коду. Це властивість важлива, якщо автентифікація з використанням геш-функції включає секретне значення. Саме секретне значення може не посилати, проте, якщо геш-функція не є однобічною, супротивник може легко розкрити секретне значення в такий спосіб. При перехопленні передачі атакуючий одержує повідомлення M и геш-код $C = H(S_{AB} || M)$. Якщо атакуючий може інвертувати геш-функцію, те, отже, він може одержати $S_{AB} || M = H^{-1} (C)$. Так як атакуючий тепер знає й M и $S_{AB} || M$, одержати S_{AB} зовсім просто.

П'ята властивість гарантує, що неможливо знайти інше повідомлення, чиє значення геш-функції збігалося б зі значенням геш-функції даного повідомлення. Це запобігає підробці автентифікатора при використанні зашифрованого геш-кода. У цьому випадку супротивник може читати повідомлення й, отже, створити його геш-код. Але так як супротивник не володіє секретним ключем, він не має можливості змінити повідомлення так, щоб одержувач цього не виявив . Якщо дана властивість не виконується, що атакує має можливість виконати наступну послідовність дій: перехопити повідомлення і його зашифрований геш-код, обчислити геш-код повідомлення, створити альтернативне повідомлення з тим же самим геш-кодом, замінити вихідне повідомлення на підроблене. Оскільки геш-коди цих повідомлень збігаються, одержувач не виявить підміни.

Геш-функція, що задовольняє першим п'яти властивостям, називається простою або слабкою геш-функцією. Якщо крім того виконується шоста властивість, то така функція називається сильною геш-функцією. Шоста властивість захищає проти класу атак, відомих як атака "день народження".

Прості геш-функції

Всі геш-функції виконуються в такий спосіб. Вхідне значення (повідомлення, файл і т.п.) розглядається як послідовність n-бітних блоків. Вхідне значення обробляється послідовно блок за блоком, і створюється m-бітне значення геш-кода.

Одним з найпростіших прикладів геш-функції є побітний XOR кожного блоку:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{ik},$$

де:

- C_i – i-ий біт геш-кода, $1 \leq i \leq n$.
- k – число n-бітних блоків входу.
- b_{ij} – i-ий біт в j-ом блоці.
- $\oplus$- операція XOR.

У результаті виходить геш-код довжини n, відомий як поздовжній надлишковий контроль. Це ефективно при випадкових збоях для перевірки цілісності даних.

Часто при використанні подібного поздовжнього надлишкового контролю для кожного блоку виконується однобітне циклічне зрушення після обчислення геш-кода. Це можна описати в такий спосіб.

1. Установити n-бітний геш-код у нуль.
2. Для кожного n-бітного блоку даних виконати наступні операції:

– зрушити циклічно поточний геш-код уліво на один біт;

– виконати операцію XOR для чергового блоку й геш-кода.

Це дасть ефект "випадковості" входу й знищить будь-яку регулярність, що є присутнім у вхідних значеннях.

Хоча другий варіант вважається більш кращим для забезпечення цілісності даних і запобігання від випадкових збоїв, він не може використовуватися для виявлення навмисних модифікацій переданих повідомлень. Знаючи повідомлення, що атакує легко може створити нове повідомлення, що має той же самий геш-код. Для цього варто підготувати альтернативне повідомлення й потім приєднати n-бітний блок, що є геш-кодом нового повідомлення, і блок, що є геш-кодом старого повідомлення.

Хоча простого XOR або ротаційного XOR (RXOR) недостатньо, якщо цілісність забезпечується тільки зашифрованим геш-кодом, а саме повідомлення не шифрується, подібна проста функція може використовуватися, коли все повідомлення й приєднаний до нього геш-код шифруються. Але й у цьому випадку варто пам'ятати про те, що подібна геш-функція не може простежити за тим, щоб при передачі послідовність блоків не змінилася. Це відбувається в силу того, що дана геш-функція визначається в такий спосіб: для повідомлення, що складає з послідовності 64-бітних блоків X_1, X_2,..., X_N, визначається геш-код З як поблочний XOR всіх блоків, що приєднується як останній блок:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N.$$

Потім все повідомлення шифрується, включаючи геш-код, у режимі CBC для створення зашифрованих блоків $Y_1, Y_2, ..., Y_{N+1}$.

По визначенню CBC маємо:

$$X_1 = IV \oplus D_K [Y_1].$$

$$X_i = Yi\text{-}1 \oplus D_K [Y_i].$$

$$X_{N+1} = Y_N \oplus D_K [Y_{N+1}].$$

Але X_{N+1} є геш-кодом:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N =$$

$$(IV \oplus D_K [Y_1]) \oplus (Y_1 \oplus D_K [Y_2]) \oplus \ldots \oplus (Y_{N-1} \oplus D_K [Y_N]).$$

Так як співмножники в попередній рівності можуть обчислюватися в будь-якому порядку, отже, геш-код не буде змінений, якщо зашифровані блоки будуть переставлені.

Первісний стандарт, запропонований NIST, використовував простий XOR, що застосовувався до 64-бітних блоків повідомлення, потім все повідомлення шифрувалося, використовуючи режим CBC.

На рисунку 3.4 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Зі схеми ми бачимо, що розроблений програмний комплекс складається з чотирьох функціональних блоків:

– Блок алгоритмів підрахунку контрольних сум (CRC).

– Блок алгоритмів шифрування.

– Блок геш-функцій.

– Блок алгоритмів генерації псевдовипадкових чисел (ГПВЧ).

У блоці алгоритмів підрахунку контрольних сум реалізовані наступні алгоритми:

– CRC32.

– XOR32bit.

– XOR16bit.

– CRC16-CCITT.

– CRC16-Standard.

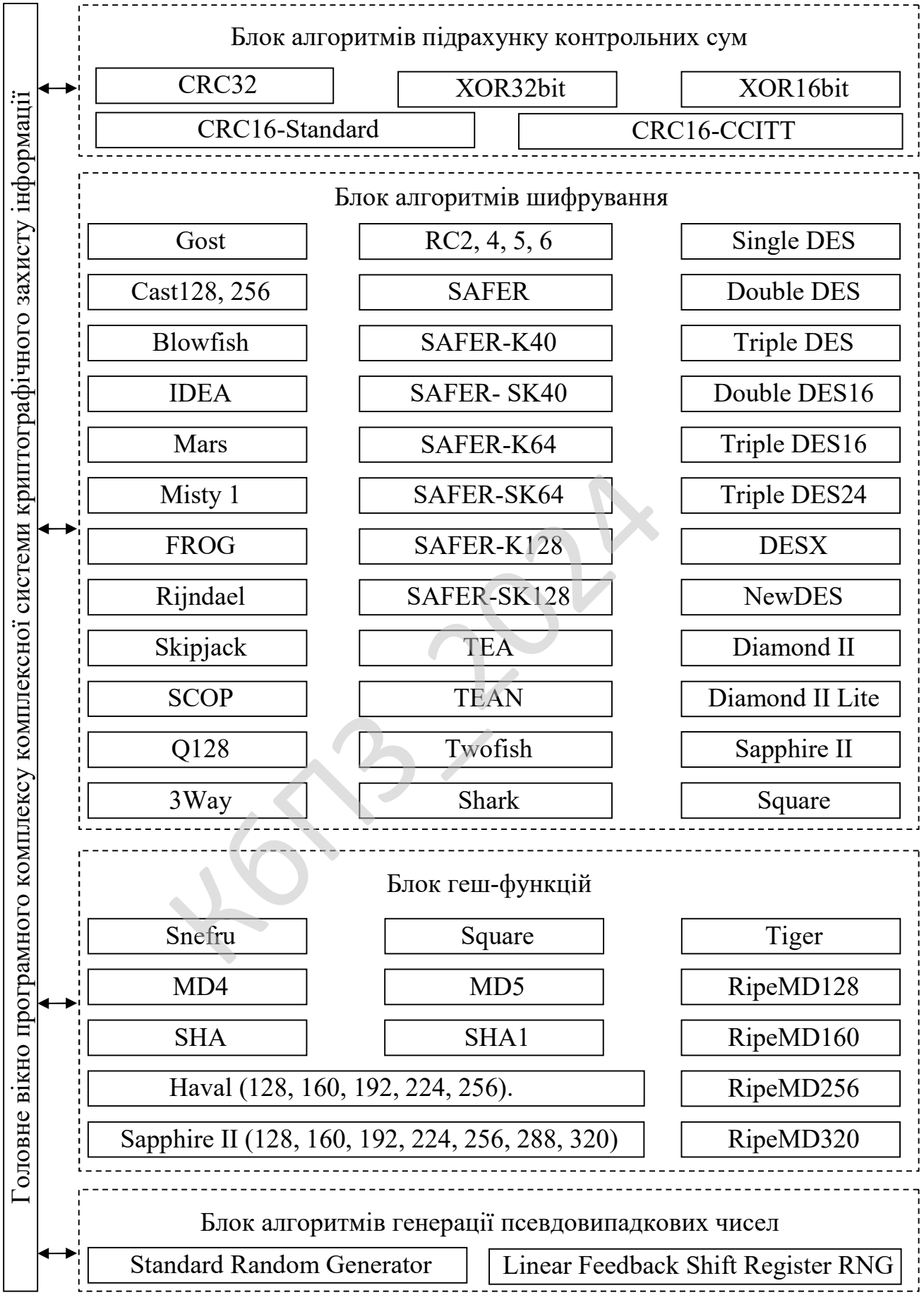

Рисунок 3.4 – Функціональна схема системи

У блоці алгоритмів шифрування реалізовані наступні криптоалгоритми:

– Gost.
– Cast128.
– Cast256.
– Blowfish.
– IDEA.
– Mars.
– Misty 1.
– RC2.
– RC4.
– RC5.
– RC6.
– FROG.
– Rijndael.
– SAFER.
– SAFER-K40.
– SAFER-SK40.
– SAFER-K64.
– SAFER-SK64.
– SAFER-K128.
– SAFER-SK128.
– TEA.
– TEAN.
– Skipjack.
– SCOP.
– Q128.
– 3Way.
– Twofish.

– Shark.
– Square.
– Single DES.
– Double DES.
– Triple DES.
– Double DES16.
– Triple DES16.
– TripleDES24.
– DESX.
– NewDES.
– Diamond II.
– Diamond II Lite.
– Sapphire II.

У блоці геш-функцій реалізовані наступні алгоритми:

– MD4.
– MD5.
– SHA (друге найменування SHS).
– SHA1.
– RipeMD128.
– RipeMD160.
– RipeMD256.
– RipeMD320.
– Haval (128, 160, 192, 224, 256).
– Snefru.
– Square.
– Tiger.
– Sapphire II (128, 160, 192, 224, 256, 288, 320).

У блоці алгоритмів генерації псевдовипадкових чисел реалізовані наступні

алгоритми:

– Standard Random Generator.

– Linear Feedback Shift Register RNG з змінним періодом з 2^{64}-1 до 2^{2032}-1.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.5. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Вибір даних для обробки.

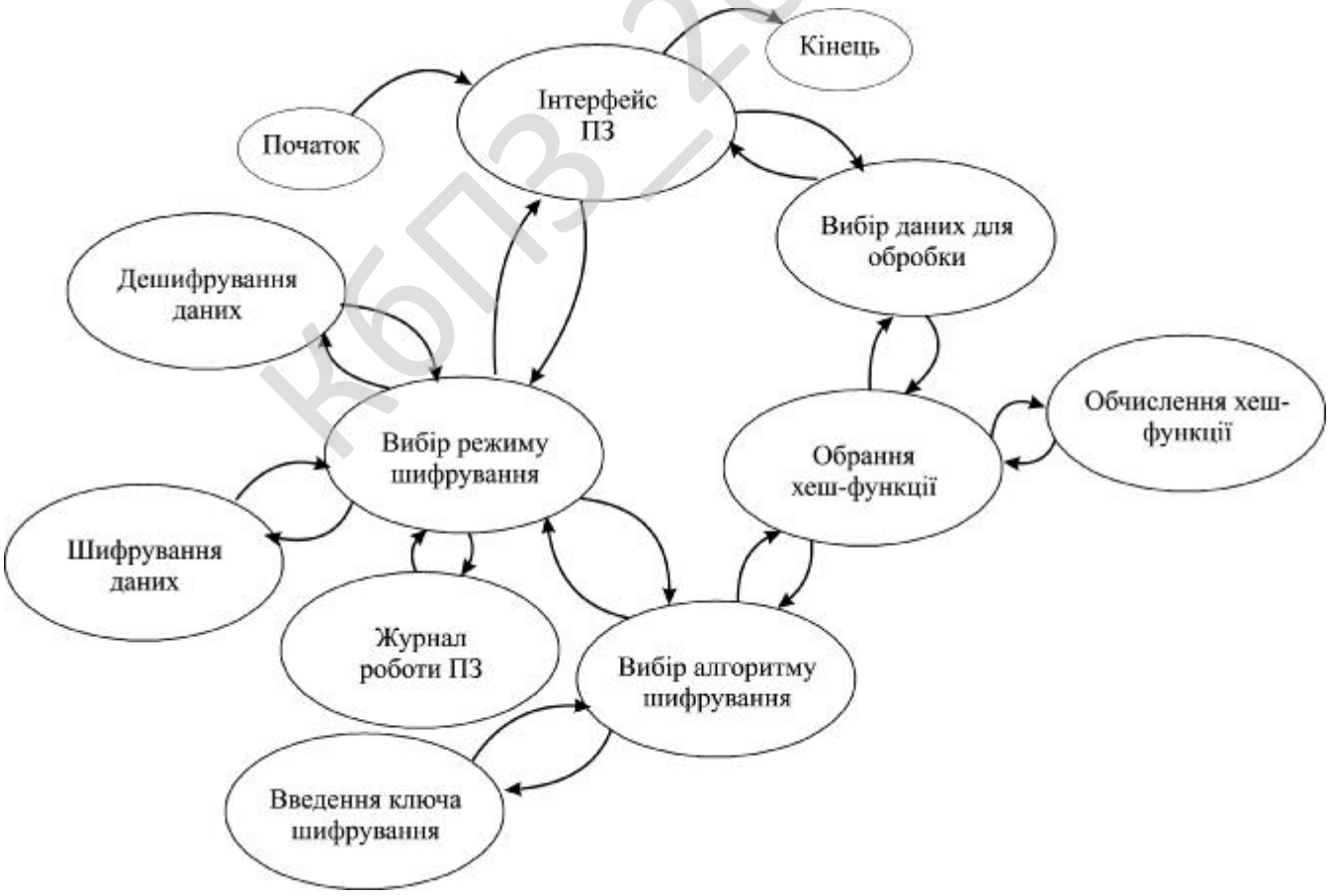

Рисунок 3.5 – Діаграма взаємодії процесів

– Обрання геш функції.

– Обчислення геш функції.

– Вибір алгоритму шифрування.

– Введення ключа шифрування.

– Вибір режиму шифрування.

– Дешифрування даних.

– Шифрування даних.

– Журнал роботи ПЗ.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

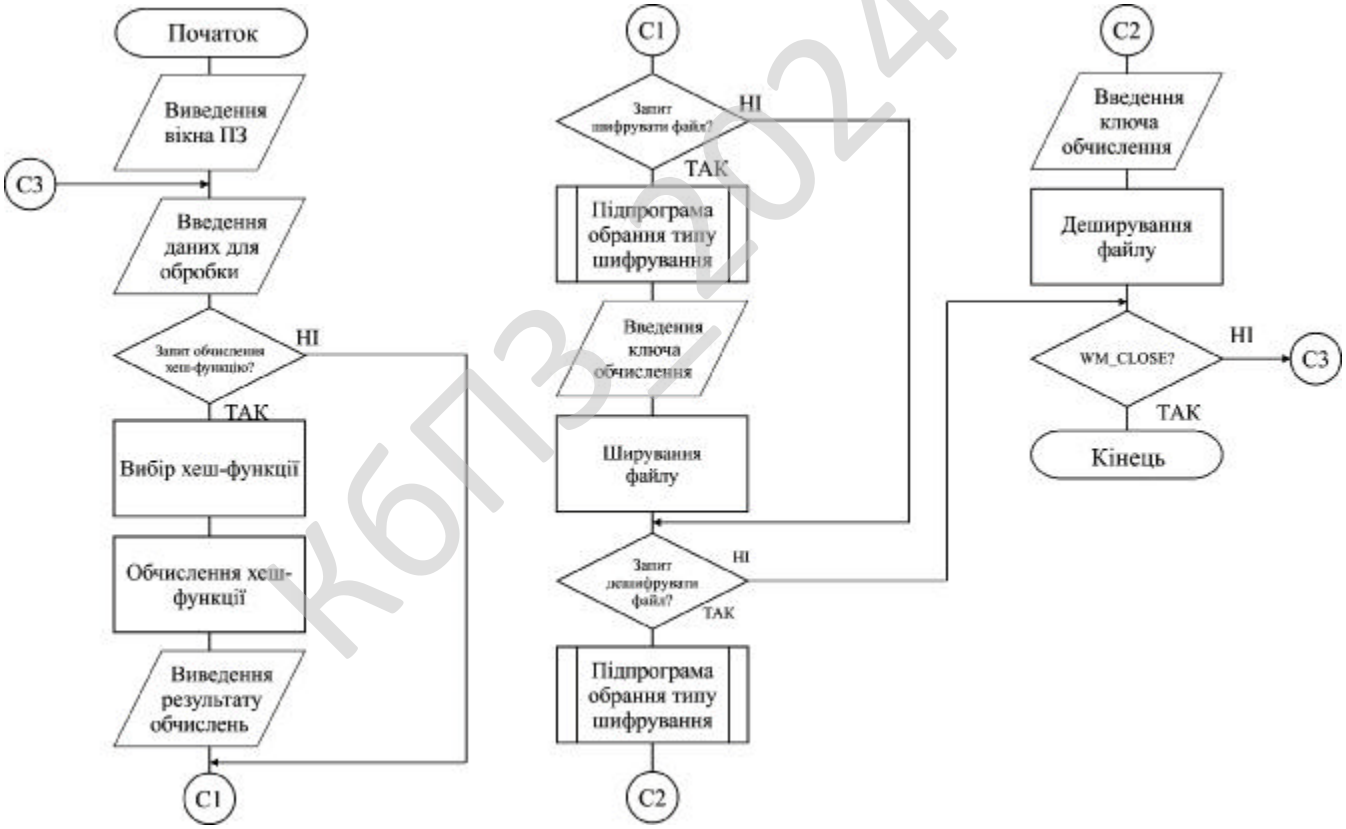

Рисунок 4.1 – Блок схема основної програми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного

циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей.

Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

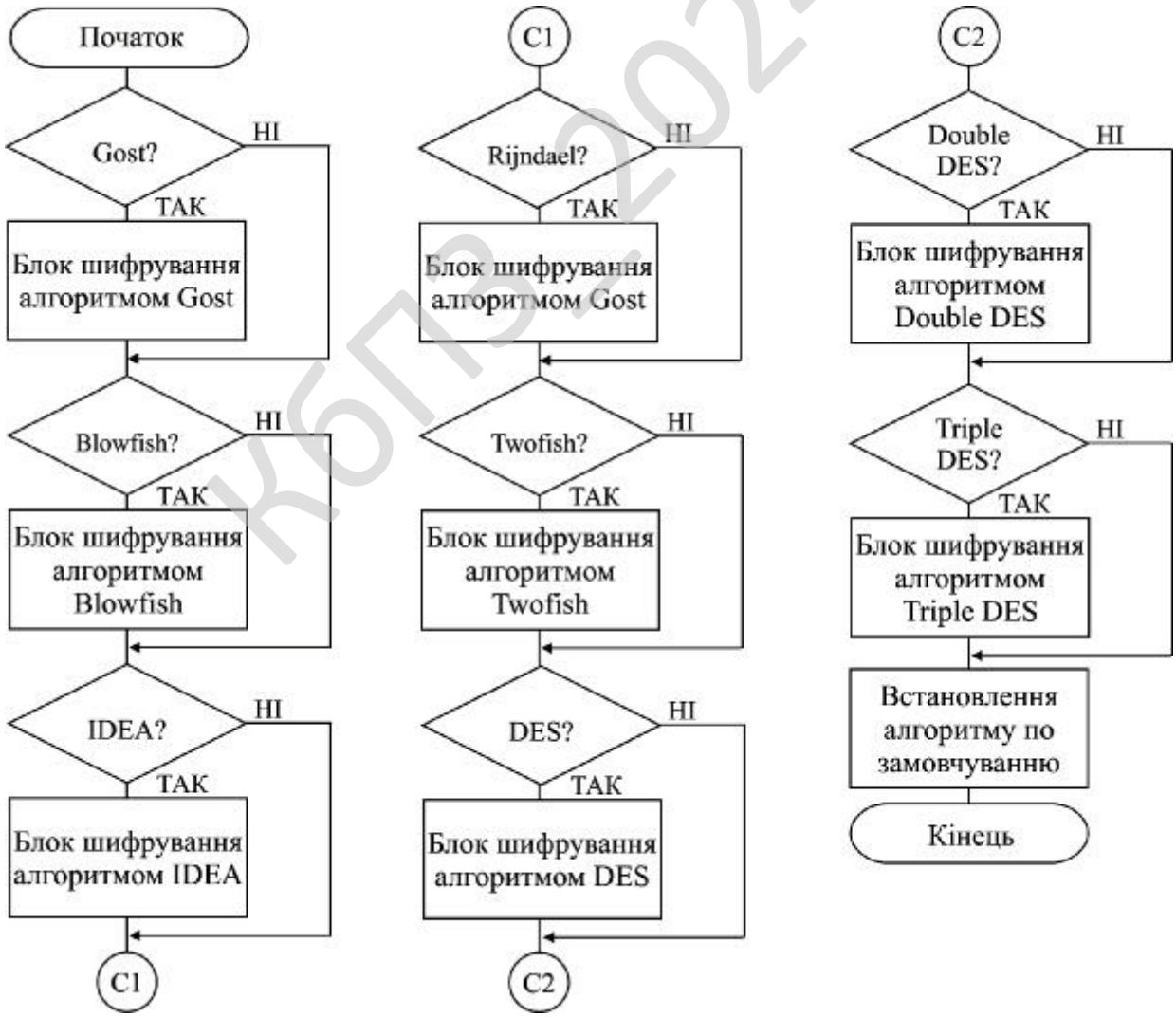

Рисунок 4.2 – Блок схема підпрограми

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності.

Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Розглянемо вихідний код, який дозволяє зареєструвати ці класи алгоритмів шифрування.

```
{$IFNDEF ManualRegisterClasses}
  RegisterCipher(TCipher_3Way, '', '');
  RegisterCipher(TCipher_Blowfish, '', '');
  RegisterCipher(TCipher_Gost, '', '');
  RegisterCipher(TCipher_IDEA, '', ' IDEA ');
  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}
```

Розглянемо вихідний код опису алгоритмів - опис алгоритмів шифрування.

Опис алгоритму SAFER_K128.

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

Опис алгоритму SAFER_SK128.

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

Опис алгоритму TEA.

```
  TCipher_TEA = class(TCipher) {Tiny Encryption Algorithm}
  private
    FRounds: Integer; {16 - 32, за замовчуванням 16 }
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    property Rounds: Integer read FRounds write SetRounds;
  end;
```

Опис алгоритму TEAN.

```
  TCipher_TEAN = class(TCipher_TEA) {Tiny Encryption Algorithm, розширена версія }
  protected
    class function TestVector: Pointer; override;
```

```
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  end;
```

Опис алгоритму SCOP.

```
  TCipher_SCOP = class(TCipher) {Потоковий шифр  в блочному режимі}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    procedure Done; override;
  end;
```

Опис алгоритму Q128.

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

Опис алгоритму 3Way.

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

Опис алгоритму Twofish.

```
  TCipher_Twofish = class(TCipher)
  protected
```

```
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
  end;
```

Опис алгоритму Shark.

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

Опис алгоритму Square.

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

Опис алгоритму ДСТ.

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

Опис алгоритму Blowfish.

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

Опис алгоритму IDEA.

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

Налаштування алгоритму SAFER.

```
  TSAFERMode = (smDefault, smK40, smK64, smK128, smStrong, smSK40, smSK64,
smSK128);
{smDefault                  Режим побудови KeyLength "Size"
якщо Size <=  5 тоді smK40 використовується
якщо Size <=  8 тоді smK64 використовується
якщо Size <= 16 тоді smK128 використовується
 smK40       SAFER K-40    Keysize рівний 40bit  ->  5 Byte
 smK64       SAFER K-64    Keysize рівний 64bit  ->  8 Byte
 smK128      SAFER K-128   KeySize рівний 128bit -> 16 Byte
```

```
 smStrong                  Режим побудови KeyLength "Size" зафіксований як
smDefault,
якщо Size <=  5 тоді smSK40 використовується
якщо Size <=  8 тоді smSK64 використовується
якщо Size <= 16 тоді smSK128 використовується
 це Defaultmode для TCipher_SAFER
 smSK40      SAFER SK-40   зафіксовано в версії для K-40 краще з Keyscheduling
 smSK64      SAFER SK-64   зафіксовано в версії для K-64 краще з Keyscheduling
 smSK128     SAFER SK-128  зафіксовано в версії для K-128 краще з Keyscheduling}
```

Опис алгоритму SAFER.

```
  TCipher_SAFER = class(TCipher) {SAFER = Secure And Fast Encryption Routine}
  private
    FRounds: Integer;
    FSAFERMode: TSAFERMode;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    procedure InitNew(const Key; Size: Integer; IVector: Pointer; SAFERMode:
TSAFERMode);
    property Rounds: Integer read FRounds write SetRounds;
  end;
```

Опис алгоритму SAFER_K40.

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

Опис алгоритму SAFER_SK40.

```
  TCipher_SAFER_SK40 = class(TCipher_SAFER_K40)
  protected
    class function TestVector: Pointer; override;
  public
```

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

Опис алгоритму SAFER_K64.

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

Опис алгоритму SAFER_SK64.

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

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 106, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти

придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. Користувач може обрати або функцію гешування, або функцію шифрування інформації. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Вибір геш-функції, яка буде використовуватися.

– Шлях до файлу, який потрібно підвергнути гешуванню.

– HEX-код, цього файлу.

– Вибір алгоритму шифрування, який буде використовуватися.

– Шлях до файлу, який потрібно зашифрувати.

– Ключ шифрування.

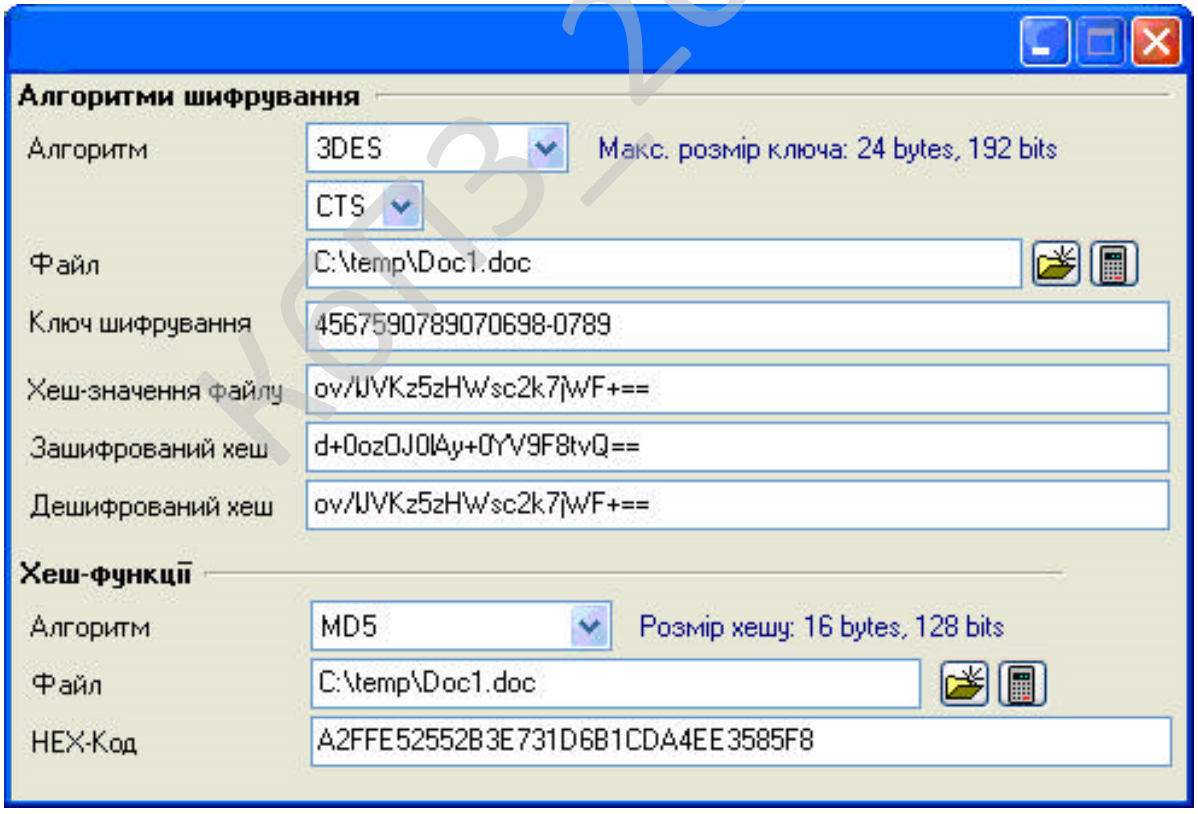

Рисунок 5.1 – Головне вікно програми

– Геш значення файлу.

– Зашифрований геш.

– Дешифрований геш.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

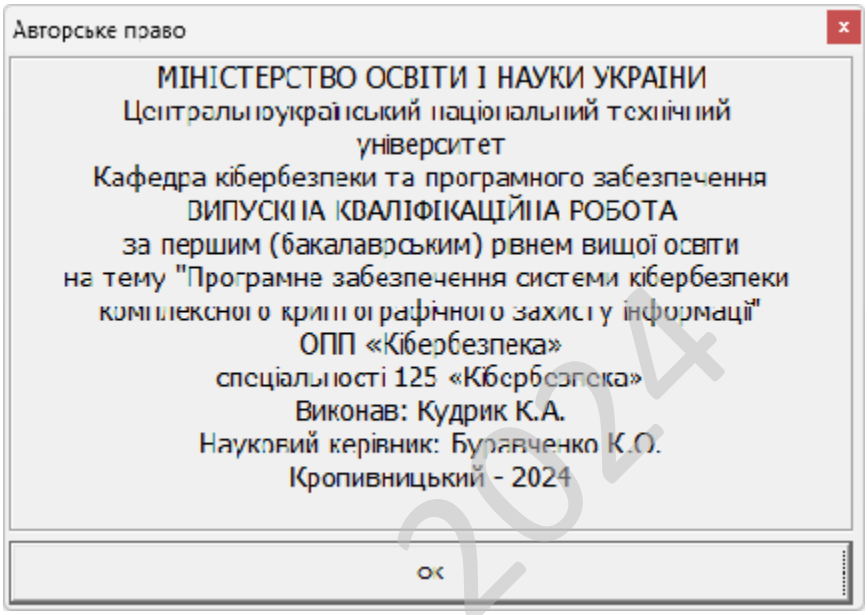

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – commercial software. Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій. Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки комплексного криптографічного захисту інформації.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем комплексного криптографічного захисту інформації.

– Досліджена система комплексного криптографічного захисту інформації.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки комплексного криптографічного захисту інформації.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання комплексного криптографічного захисту інформації.

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 комплексного криптографічного захисту інформації. Це

дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

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. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

2. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

25. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

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

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

31. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of

Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

32. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки комплексного криптографічного захисту інформації.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 138-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки комплексного криптографічного захисту інформації.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки комплексного криптографічного захисту інформації;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище 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 – 70 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки комплексного криптографічного захисту інформації

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 65

Літера: РП

Кропивницький – 2024 року

Файл Main.pas – основна програма

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons, XPMan;

type
  TMainForm = class(TForm)
    PHash: TPanel;
    Label1: TLabel;
    Algorithm: TLabel;
    CBHash: TComboBox;
    EHashFile: TEdit;
    Label2: TLabel;
    BtnHashFile: TBitBtn;
    BtnCalcHash: TBitBtn;
    OpenDialog: TOpenDialog;
    LhashInfo: TLabel;
    Label3: TLabel;
    EDigest: TEdit;
    PCipher: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    CBCipher: TComboBox;
    Label6: TLabel;
    ECipherFile: TEdit;
    BtnCipherFile: TBitBtn;
    BtnCalcCipher: TBitBtn;
    LCipherInfo: TLabel;
    EPassword: TEdit;
    Label7: TLabel;
    Label8: TLabel;
    EHashInput: TEdit;
    Label9: TLabel;
    EHashENC: TEdit;
    Label10: TLabel;
    EhashDEC: TEdit;
    CBMode: TComboBox;
    XPManifest1: TXPManifest;
    SpeedButton1: TSpeedButton;
    Bevel1: TBevel;
    Bevel2: TBevel;
    procedure FormCreate(Sender: TObject);
    procedure BtnHashFileClick(Sender: TObject);
    procedure BtnCalcHashClick(Sender: TObject);
    procedure BtnCipherFileClick(Sender: TObject);
    procedure BtnCalcCipherClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  MainForm: TMainForm;

implementation

uses about;

{$R *.DFM}
```

```
{Імпортуємо прототипи з DEC1.DLL}

const
  DEC1DLL = 'DEC1.DLL';

type
  TCipherHandle = Integer;
  THashHandle   = Integer;
  TEnumProc     = function(const Name: PChar; ID, MaxKeySize: Integer; Data:
Pointer): Bool; stdcall;

const
{Режими шифрування для Cipher_Create()}
  cm_CTS        = 0;
  cm_CBC        = 1;
  cm_CFB        = 2;
  cm_OFB        = 3;
  cm_ECB        = 4;

function Cipher_GetID(Name: PChar): Integer; stdcall; external DEC1DLL;
function Cipher_Create(ID, Mode: Integer): TCipherHandle; stdcall; external
DEC1DLL;
function Cipher_Delete(Handle: TCipherHandle): Integer; stdcall; external
DEC1DLL;
function Cipher_Encode(Handle: TCipherHandle; Source, Dest: PChar; Size:
Integer): Integer; stdcall; external DEC1DLL;
function Cipher_Decode(Handle: TCipherHandle; Source, Dest: PChar; Size:
Integer): Integer; stdcall; external DEC1DLL;
function Cipher_Init(Handle: TCipherHandle; Key: Pointer; KeyLen: Integer;
IVector: Pointer): Integer; stdcall; external DEC1DLL;
function Cipher_InitKey(Handle: TCipherHandle; Key: PChar; IVector: Pointer):
Integer; stdcall; external DEC1DLL;
function Cipher_Done(Handle: TCipherHandle): Integer; stdcall; external DEC1DLL;
function Cipher_Protect(Handle: TCipherHandle): Integer; stdcall; external
DEC1DLL;
function Cipher_GetMaxKeySize(Handle: TCipherHandle): Integer; stdcall; external
DEC1DLL;
function Cipher_SetHash(Handle: TCipherHandle; Hash_ID: Integer): Integer;
stdcall; external DEC1DLL;
function Cipher_GetHash(Handle: TCipherHandle): Integer; stdcall; external
DEC1DLL;
procedure Cipher_EnumNames(Proc: TEnumProc; UserData: Pointer); stdcall;
external DEC1DLL;

function Hash_GetID(Name: PChar): Integer; stdcall; external DEC1DLL;
function Hash_Create(ID: Integer): THashHandle; stdcall; external DEC1DLL;
function Hash_Delete(Handle: THashHandle): Integer; stdcall; external DEC1DLL;
function Hash_Init(Handle: THashHandle): Integer; stdcall; external DEC1DLL;
function Hash_Done(Handle: THashHandle; Digest: PChar; DigestSize: Integer):
Integer; stdcall; external DEC1DLL;
function Hash_Update(Handle: THashHandle; Source: PChar; SourceLen: Integer):
Integer; stdcall; external DEC1DLL;
function Hash_GetMaxDigestSize(Handle: THashHandle): Integer; stdcall; external
DEC1DLL;
function Hash_CalcFile(ID: Integer; FileName, Digest: PChar; MaxDigestLen:
Integer): Integer; stdcall; external DEC1DLL;
procedure Hash_EnumNames(Proc: TEnumProc; UserData: Pointer); stdcall; external
DEC1DLL;

function StrToBase64(Dest, Source: PChar; Len, MaxLen: Integer): Integer;
stdcall; external DEC1DLL;
function Base64ToStr(Dest, Source: PChar; Len, MaxLen: Integer): Integer;
stdcall; external DEC1DLL;
function StrToBase16(Dest, Source: PChar; Len, MaxLen: Integer): Integer;
stdcall; external DEC1DLL;
function Base16ToStr(Dest, Source: PChar; Len, MaxLen: Integer): Integer;
stdcall; external DEC1DLL;

{Кінець імпорту}
```

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

  function EnumHash(const Name: PChar; ID, MaxKeySize: Integer; Combo:
TComboBox): Bool; stdcall;
  {ітерації з ID=0 до HashCount-1}
  begin
    Result := True;
    Combo.Items.AddObject(Name, Pointer(MaxKeySize));
  end;

  function EnumCipher(const Name: PChar; ID, MaxKeySize: Integer; Combo:
TComboBox): Bool; stdcall;
  begin
    Result := True;
    Combo.Items.AddObject(Name, Pointer(MaxKeySize));
  end;

begin
  EHashFile.Text := ParamStr(0);
  ECipherFile.Text := ParamStr(0);

  Hash_EnumNames(@EnumHash, CBHash);
  CBHash.ItemIndex := 0;
  BtnCalcHashClick(nil);

  Cipher_EnumNames(@EnumCipher, CBCipher);
  CBCipher.ItemIndex := 0;
  CBMode.ItemIndex := 0;
  BtnCalcCipherClick(nil);
end;

procedure TMainForm.BtnHashFileClick(Sender: TObject);
begin
  if OpenDialog.Execute then
  begin
    EHashFile.Text := OpenDialog.FileName;
    BtnCalcHashClick(nil);
  end;
end;

procedure TMainForm.BtnCalcHashClick(Sender: TObject);
var
  Handle: THashHandle;
  Len: Integer;
  S: TFileStream;
  Buffer: array[0..1023] of Char;
  Digest: String;
begin
  if FileExists(EHashFile.Text) and (CBHash.ItemIndex >= 0) then
  begin
    Len := Integer(CBHash.Items.Objects[CBHash.ItemIndex]);
    LHashInfo.Caption := Format('Розмір хешу: %d bytes, %d bits', [Len, Len *
8]);
    SetLength(Digest, Len);

    Handle := Hash_Create(CBHash.ItemIndex);
    if Hash_Init(Handle) = 0 then
    try
      S := TFileStream.Create(EHashFile.Text, fmOpenRead or fmShareDenyNone);
      try
        repeat
          Len := S.Read(Buffer, Sizeof(Buffer));
          Hash_Update(Handle, Buffer, Len);
        until Len <= 0;
        Hash_Done(Handle, PChar(Digest), Length(Digest));

        StrToBase16(Buffer, PChar(Digest), Length(Digest), SizeOf(Buffer));
        EDigest.Text := StrPas(Buffer);
```

```
      finally
        S.Free;
      end;
    finally
      Hash_Delete(Handle);
    end else EDigest.Text := 'Помилка введення';
  end;
end;

procedure TMainForm.BtnCipherFileClick(Sender: TObject);
begin
  if OpenDialog.Execute then
  begin
    ECipherFile.Text := OpenDialog.FileName;
    BtnCalcCipherClick(nil);
  end;
end;

procedure TMainForm.BtnCalcCipherClick(Sender: TObject);

  procedure HashBase64(const FileName: String; Output: TEdit);
  var
    Digest: String;
    Len: Integer;
  begin
    SetLength(Digest, 1024);
    Len := Hash_CalcFile(CBHash.ItemIndex, PChar(FileName), PChar(Digest),
Length(Digest));
    if (Len > 0) and (StrToBase64(PChar(Digest), PChar(Digest), Len,
Length(Digest)) = 0) then
      Output.Text := Digest
    else Output.Text := 'Error';
  end;

var
  Len: Integer;
  Handle: TCipherHandle;
  S,D: TFileStream;
  Buffer: array[0..1023] of Char;
begin
  if FileExists(ECipherFile.Text) and (CBCipher.ItemIndex >= 0) then
  try
    Screen.Cursor := crHourGlass;

    Len := Integer(CBCipher.Items.Objects[CBCipher.ItemIndex]);
    LCipherInfo.Caption := Format('Макс. розмір ключа: %d bytes, %d bits', [Len,
Len * 8]);

{створюємо шифр}
    Handle := Cipher_Create(CBCipher.ItemIndex, CBMode.ItemIndex);
{встановлюємо ключ шифрування хеш функції  SHA1}
    Cipher_SetHash(Handle, CBHash.ItemIndex);
    try
{встановлюємо фази шифрування}
      Cipher_InitKey(Handle, PChar(EPassword.Text), nil);
{open Source & Desfile, read in, encode}
      S := nil;
      D := nil;
      try
        S := TFileStream.Create(ECipherFile.Text, fmOpenRead or
fmShareDenyNone);
        D := TFileStream.Create(ChangeFileExt(ParamStr(0), '.ENC'), fmCreate);
        repeat
          Len := S.Read(Buffer, SizeOf(Buffer));
          Cipher_Encode(Handle, Buffer, Buffer, Len);
          D.Write(Buffer, Len);
        until Len <= 0;
      finally
        Cipher_Protect(Handle);
```

```
        S.Free;
        D.Free;
      end;
{і тепер назад, дешифруємо}
      Cipher_InitKey(Handle, PChar(EPassword.Text), nil);
      S := nil;
      D := nil;
      try
        S := TFileStream.Create(ChangeFileExt(ParamStr(0), '.ENC'), fmOpenRead
or fmShareDenyNone);
        D := TFileStream.Create(ChangeFileExt(ParamStr(0), '.DEC'), fmCreate);
        repeat
          Len := S.Read(Buffer, SizeOf(Buffer));
          Cipher_Decode(Handle, Buffer, Buffer, Len);
          D.Write(Buffer, Len);
        until Len <= 0;
      finally
        Cipher_Protect(Handle);
        S.Free;
        D.Free;
      end;
    finally
      Cipher_Delete(Handle);
    end;

{перевіряємо роботу}
    HashBase64(ECipherFile.Text, EHashInput);
    HashBase64(ChangeFileExt(ParamStr(0), '.ENC'), EHashENC);
    HashBase64(ChangeFileExt(ParamStr(0), '.DEC'), EHashDEC);
    if EHashInput.Text <> EHashDEC.Text then EHashDEC.Color := clRed
      else EHashDEC.Color := clWindow;
  finally
    Screen.Cursor := crDefault;
  end;
end;

procedure TMainForm.SpeedButton1Click(Sender: TObject);
begin
Form1.Show;
end;

end.
```

Файл Cipher.pas – криптографічні алгоритми

```
unit Cipher;

interface

{$I VER.INC}

uses SysUtils, Classes, DECUtil, Hash;

const {ErrorCode's for ECipherException}
  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 використовує cmECB, cmOFB}
  errCantCalc       = 5;

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

{Усі класи шифрування у данній бібліотеці мають добрі параметри}
  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;  {Потоковий шифр}
  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
  cmCBC     Cipher Block Chaining
  cmCFB     K-bit Cipher Feedback
  cmOFB     K-bit Output Feedback
  cmECB *   Electronic Codebook

  cmCTSMAC  Build a Message Authentication Code у cmCTS режимі
  cmCBCMAC  Build a CBC-MAC
  cmCFBMAC  Build a 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;
{повинно анулюватися}
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
virtual;
    class function TestVector: Pointer; virtual;
{анулює TProtection методи}
    procedure CodeInit(Action: TPAction); override;
    procedure CodeDone(Action: TPAction); override;
    procedure CodeBuf(var Buffer; const BufferSize: Integer; Action: TPAction);
override;
{функція шифрування , повинно анулюватися}
    procedure Encode(Data: Pointer); virtual;
{ функція дешифрування, повинно анулюватися}
    procedure Decode(Data: Pointer); virtual;
{особисті дані користувача й буфер}
    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; override;
    class function MaxKeySize: Integer;
{простий тест на коректну роботу}
    class function SelfTest: Boolean;
{ініціалізує вікно шифрування}
    procedure Init(const Key; Size: Integer; IVector: Pointer); virtual;
    procedure InitKey(const Key: String; IVector: Pointer);
{сбрасує Feedbackregister с IVector}
    procedure Done; virtual;
{захищає таємні Data's, Feedback, Buffer, Vector etc.}
    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);

{розраховуєа MAC, Message Authentication Code, використовується в
 cmCBCMAC, cmCTSMAC, cmCFBMAC Modes
 cmCBC, cmCTS, cmCFB Modes }
    function CalcMAC(Format: Integer): String;

{Режим шифрування = cmXXX}
    property Mode: TCipherMode read FMode write FMode;
{поточний Hash-Object, з InitKey()}
    property Hash: THash read GetHash;
{Клас Hash-Object}
    property HashClass: THashClass read FHashClass write SetHashClass;
{максимальний KeySize та BufSize (Size of Feedback, Buffer та Vector}
```

```
    property KeySize: Integer read FKeySize;
    property BufSize: Integer read FBufSize;

{Init() викликається}
    property Initialized: Boolean index 1 read GetFlag write SetFlag;
{актуальний IVector, BufSize Bytes long}
    property Vector: Pointer read FVector;
{ Feedback register, BufSize Bytes long}
    property Feedback: Pointer read FFeedback;
{ Key у функції InitKey }
    property HasHashKey: Boolean index 0 read GetFlag;
  end;

// Опис алгоритмів шифрування

// Алгоритм ГОСТ

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

// Алгоритм Blowfish

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

// Алгоритм IDEA

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

// Вибір режиму алгоритму SAFER

  TSAFERMode = (smDefault, smK40, smK64, smK128, smStrong, smSK40, smSK64,
smSK128);
{smDefault                 Режим побудови KeyLength "Size"
                           якщо Size <=  5 тоді smK40 використовується
```

```
                          якщо Size <=  8 тоді smK64 використовується
                          якщо Size <= 16 тоді smK128 використовується
 smK40       SAFER K-40   Keysize рівний 40bit  ->  5 Byte
 smK64       SAFER K-64   Keysize рівний 64bit  ->  8 Byte
 smK128      SAFER K-128  KeySize рівний 128bit -> 16 Byte
 smStrong                 Режим побудови KeyLength "Size" зафіксований як
smDefault,
                          якщо Size <=  5 тоді smSK40 використовується
                          якщо Size <=  8 тоді smSK64 використовується
                          якщо Size <= 16 тоді smSK128 використовується
                          це Defaultmode для TCipher_SAFER
 smSK40      SAFER SK-40  зафіксовано в версії для K-40 краще з Keyscheduling
 smSK64      SAFER SK-64  зафіксовано в версії для K-64 краще з Keyscheduling
 smSK128     SAFER SK-128 зафіксовано в версії для K-128 краще з Keyscheduling}

// Алгоритм SAFER

  TCipher_SAFER = class(TCipher) {SAFER = Secure And Fast Encryption Routine}
  private
    FRounds: Integer;
    FSAFERMode: TSAFERMode;
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    procedure InitNew(const Key; Size: Integer; IVector: Pointer; SAFERMode:
TSAFERMode);
    property Rounds: Integer read FRounds write SetRounds;
  end;

// Алгоритм SAFER_K40

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

// Алгоритм SAFER_SK40

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

// Алгоритм SAFER_K64

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

// Алгоритм SAFER_SK64

  TCipher_SAFER_SK64 = class(TCipher_SAFER_K64)
```

```
  protected
    class function TestVector: Pointer; override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
  end;

// Алгоритм SAFER_K128

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

// Алгоритм SAFER_SK128

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

// Алгоритм TEA

  TCipher_TEA = class(TCipher) {Tiny Encryption Algorithm}
  private
    FRounds: Integer; {16 - 32, за замовчуванням 16 }
    procedure SetRounds(Value: Integer);
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    property Rounds: Integer read FRounds write SetRounds;
  end;

// Алгоритм TEAN

  TCipher_TEAN = class(TCipher_TEA) {Tiny Encryption Algorithm, розширена версія
}
  protected
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  end;

// Алгоритм SCOP

  TCipher_SCOP = class(TCipher) {Потоковий шифр  в блочному режимі}
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
    procedure Done; override;
  end;

// Алгоритм Q128
```

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

// Алгоритм 3Way

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

// Алгоритм Twofish

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

// Алгоритм Shark

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

// Алгоритм Square

  TCipher_Square = class(TCipher)
  protected
    class procedure GetContext(var ABufSize, AKeySize, AUserSize: Integer);
override;
    class function TestVector: Pointer; override;
    procedure Encode(Data: Pointer); override;
    procedure Decode(Data: Pointer); override;
  public
    procedure Init(const Key; Size: Integer; IVector: Pointer); override;
  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;
{встановлюємо в True raises Exception коли Розмір ключа дуже великий, (Method
Init())
 Та обрізаємо Key, коли 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; {маємо модифікацію testvectors ?}
{ми повинні використовувати ClassName as Key }
  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; {Обрізаємо великий Keys}
  Init(Hash.DigestKey^, I, IVector);
  EncodeBuffer(Hash.DigestKey^, Hash.DigestKey^, Hash.DigestKeySize);
  Done;
  SetFlag(0, True);
end;

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

procedure TCipher.Protect;
begin
  SetFlag(0, False);
  Initialized := False;
  FillChar(FVector^, FBufSize, $AA);
  FillChar(FFeedback^, FBufSize, $AA);
  FillChar(FBuffer^, FBufSize, $AA);
  FillChar(FUser^, FUserSize, $AA);

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

  FillChar(FVector^, FBufSize, $FF);
  FillChar(FFeedback^, FBufSize, $FF);
  FillChar(FBuffer^, FBufSize, 0);
```

```
  FillChar(FUser^, FUserSize, 0);
end;

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

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

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

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

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

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

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

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

procedure TCipher.DecodeFile(const Source, Dest: String);
```

```
begin
  InternalCodeFile(Source, Dest, False);
end;

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

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

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

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

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

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

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

procedure TCipher.CodeInit(Action: TPAction);
begin
  if not Initialized then
    RaiseCipherException(errNotInitialized, Format(sNotInitialized,
[ClassName]));
{  if (Mode in [cmCBCMAC, cmCTSMAC, cmCFBMAC]) <> (Action = paCalc) then
    RaiseCipherException(errCantCalc, Format(sCantCalc, [ClassName]));}
  if Action <> paCalc then
    if Action <> paWipe then Done
      else RndXORBuffer(RndTimeSeed, FFeedback^, FBufSize);
  inherited CodeInit(Action);
end;
```

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

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

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

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

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

type
  PCipherRec = ^TCipherRec;
  TCipherRec = packed record
                  case Integer of
                    0: (X: array[0..7] of Byte);
```

```
                    1: (A, B: LongWord);
                end;

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

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

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

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

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

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

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

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

        MOV    ESI,[EAX].TCipher_Blowfish.FUser

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

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

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

```
        XOR     EDI,EDI

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

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

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

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

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

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

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

        POP     EBP
        POP     EBX
        POP     ESI
        POP     EDI
end;

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

        MOV     ESI,[EAX].TCipher_Blowfish.FUser

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

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

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

        MOV     EDI,16

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

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

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

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

        DEC     EDI
        JNZ     @@1

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

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

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

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

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

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

        BSWAP   EBX                // CPU >= 486
        BSWAP   EBP

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

        MOV     EAX,EBX
        SHR     EBX,16
        MOVZX   ECX,BH     // це прискорює для AMD Chips,
//        MOV     CL,BH    // це прискорює для PII's
        MOV     EDX,[ESI + ECX * 4 + 1024 * 0]
        MOVZX   ECX,BL
//        MOV     CL,BL
        ADD     EDX,[ESI + ECX * 4 + 1024 * 1]
```

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

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

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

        BSWAP  EBX
        BSWAP  EBP

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

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

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

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

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

```
        BSWAP  EBX
        BSWAP  EBP

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

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

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

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

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

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

        BSWAP  EBX
        BSWAP  EBP

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

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

```
  Dec(PInteger(P));
  PCipherRec(Data).A := SwapInteger(B xor P[0]);
  PCipherRec(Data).B := SwapInteger(A);
end;
{$ENDIF}

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

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

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

class function TCipher_IDEA.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,065h,0CAh,0D8h,043h,0E7h,099h,093h
         DB    0EDh,041h,0EAh,048h,0FDh,066h,050h,094h
         DB    0A2h,025h,06Dh,0D7h,0B1h,0D0h,09Ah,023h
         DB    03Dh,0D2h,0E8h,0ECh,0C9h,045h,07Fh,07Eh
end;

function IDEAMul(X, Y: LongWord): LongWord; assembler; register;
asm
     AND    EAX,0FFFFh
```

```
      JZ      @@1
      AND     EDX,0FFFFh
      JZ      @@1
      MUL     EDX
      MOV     ECX,EAX
      MOV     EDX,EAX
      SHR     EDX,16
      SUB     EAX,EDX
      CMP     AX,CX
      JNA     @@2
      INC     EAX
@@2:  RET
@@1:  MOV     ECX,1
      SUB     ECX,EAX
      SUB     ECX,EDX
      MOV     EAX,ECX
end;

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

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

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

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

  function IDEAInv(X: Word): Word;
  var
```

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

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

```
end;

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

procedure TCipher_SAFER.SetRounds(Value: Integer);
begin
  if (Value < 4) or (Value > 13) then
    case FSaferMode of  {Визначений раунд }
      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);
{Установка ключа A}
    I := Size;
    if I > 8 then I := 8;
    Move(Key, K[False], I);
{Установка ключа для K-40, SK-40}
    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;
{Установка KeyData}
    Move(K[True], D^, 8);
    Inc(D, 8);

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

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

begin
  InitBegin(Size);
  FSAFERMode := SAFERMode;
  if SAFERMode = smDefault then
```

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

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

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

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

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

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

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

class function TCipher_SAFER_K64.TestVector: Pointer;
asm
         MOV   EAX,OFFSET @Vector
         RET
@Vector: DB    08Ch,0B2h,032h,0F0h,00Eh,0C2h,0DAh,0CBh
         DB    039h,008h,02Dh,05Ch,093h,0FFh,0CEh,0F3h
         DB    08Fh,01Fh,0B7h,02Ch,0C5h,0C7h,0A7h,0E9h
         DB    089h,0BEh,061h,08Bh,000h,0E6h,09Fh,00Eh
end;

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

```
end;

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

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

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

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

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

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

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

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

const
  TEA_Delta = $9E3779B9;

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

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

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

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

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

         MOV   ESI,[EAX].TCipher_TEA.FUser  // Користувач
         MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди

@@1:     ADD   EDI,TEA_Delta

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

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

         XOR   EAX,EBP
         ADD   EDX,EAX

         DEC   ECX
         JNZ   @@1

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

         POP   EBP
         POP   EBX
```

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

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

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

          MOV   ESI,[EAX].TCipher_TEA.FUser  // Користувач
          MOV   EDI,TEA_Delta
          MOV   ECX,[EAX].TCipher_TEA.FRounds  // Раунди
          IMUL  EDI,ECX

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

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

          DEC   ECX
          JNZ   @@1

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

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

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

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

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

procedure TCipher_TEAN.Decode(Data: Pointer);
var
  I,Sum,X,Y: LongWord;
  K: PIntArray;
begin
  Sum := TEA_Delta * LongWord(FRounds);
  X := PTEARec(Data).A;
  Y := PTEARec(Data).B;
```

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

const
  SCOP_SIZE = 32; {is the Maximum}

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 використовується для <= CPU 386

procedure FindVirtualMethodAndChange(AClass: TClass; MethodAddr, NewAddress:
Pointer);
// MethodAddr повинно явно існувати
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 використовується 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}
{$IFNDEF ManualRegisterClasses}
  RegisterCipher(TCipher_3Way, '', '');
  RegisterCipher(TCipher_Blowfish, '', '');
  RegisterCipher(TCipher_Gost, '', '');
  RegisterCipher(TCipher_IDEA, '', 'Не для комерційного використання');
  RegisterCipher(TCipher_Q128, '', '');
  RegisterCipher(TCipher_SAFER_K40, 'SAFER-K40', '');
  RegisterCipher(TCipher_SAFER_SK40, 'SAFER-SK40', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K64, 'SAFER-K64', '');
  RegisterCipher(TCipher_SAFER_SK64, 'SAFER-SK64', 'Keyscheduling');
  RegisterCipher(TCipher_SAFER_K128, 'SAFER-K128', '');
  RegisterCipher(TCipher_SAFER_SK128, 'SAFER-SK128', 'Keyscheduling');
  RegisterCipher(TCipher_SCOP, '', '');
  RegisterCipher(TCipher_Shark, '', '');
  RegisterCipher(TCipher_Square, '', '');
  RegisterCipher(TCipher_TEA, 'TEA', '');
  RegisterCipher(TCipher_TEAN, 'TEA extended', '');
  RegisterCipher(TCipher_Twofish, '', '');
{$ENDIF}
finalization
{$IFDEF VER_D3H}
  RemoveModuleUnloadProc(ModuleUnload);
{$ENDIF}
  FCipherList.Free;
  FCipherList := nil;
end.
```

Файл about.pas – довідка

```
unit about;

interface

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

type
  TForm1 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

procedure TfmAbout.FormCreate(Sender: TObject);
begin
Label1.Caption:='БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА';
Label2.Caption:='на тему:';
Label3.Caption:=' Програмне забезпечення системи кібербезпеки комплексного
криптографічного захисту інформації ';
Label4.Caption:=' ';
Label5.Caption:='Керівник: Буравченко К.О.';
Label6.Caption:='Розробив: студент Кудрик Кирил Андрійович';
Label7.Caption:='гр. КБ-20ПЗ';
Label8.Caption:='м. Кропивницький 2024';
end;

end.
```