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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

Фадєєву Максиму Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів*

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

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

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

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

6. Консультанти розділів роботи

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Фадєєв М.О.
(прізвище та ініціали)

АНОТАЦІЯ

Фадєєв М.О. Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Метою розробки є дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Об'єктом дослідження є процес з використанням мультиваріантного центру реалізації криптоалгоритмів.

Предметом дослідження є методи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

Результат роботи – програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

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

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, захист інформації

ABSTRACT

Fadieiev M.O. Research and software implementation of the system using the multivariate center for the implementation of crypto-algorithms. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for a system using a multivariate center for the implementation of crypto-algorithms.

The purpose of the development is the research and software implementation of the system using the multivariate center for the implementation of crypto-algorithms.

The object of research is a process using a multivariate center for the implementation of crypto-algorithms.

The subject of research are methods using a multivariate center for the implementation of crypto-algorithms.

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

The result of the work is the software implementation of the system using the multivariate center for the implementation of crypto-algorithms.

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

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

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

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, information protection

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. Останнім часом досить широке поширення одержали різні програмно-апаратні системи захисту інформації, призначені для шифрування даних, що зберігаються на жорстких дисках. Як приклади можна привести відомі PGPdisk і BestCrypt, StrongDisk, Zdisk і Zserver від SecurIT, і т.д.

Крім цього, існують вартим особняком системи шифрування окремих файлів і каталогів, найбільш відома й розповсюджена з яких – EFS (Encrypted File System), що входить до складу MS Windows, починаючи з Windows 2000.

Всі ці системи відрізняються друг від друга способом шифрування, алгоритмами, можливостями й т.д. настільки, що потенційний користувач таких систем часом губиться, і не завжди може зрозуміти, які саме можливості надає та або інша система, і навіщо йому все це потрібно.

Незважаючи на масу розходжень, всі сучасні системи шифрування даних працюють за принципом «прозорого» шифрування. Суть цього принципу полягає в тому, що шифрування даних не є окремою операцією, що повинен виконувати користувач у процесі роботи, а здійснюється одночасно з роботою користувача, автоматично, при читанні й записі даних. Користувач тільки повинен включити шифрування, ввівши при цьому якийсь пароль або ключ шифрування.

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.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

– Огляд існуючих систем з використанням мультиваріантного центру реалізації криптоалгоритмів.

– Дослідження системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

– Програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Об'єктом дослідження є процес з використанням мультиваріантного центру реалізації криптоалгоритмів.

Предметом дослідження є методи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

– Удосконалено метод з використанням мультиваріантного центру реалізації криптоалгоритмів.

– Розроблено вітчизняний продукт з використанням мультиваріантного центру реалізації криптоалгоритмів, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення призначене для захисту від несанкціонованого доступу до файлів і каталогів будь-якого формату на жорстких дисках, дискетах і інших носіях 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.

1.2 Область застосування

Система з використанням мультиваріантного центру реалізації криптоалгоритмів 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.

– WEB-додатки й інформаційні портали.

– Архівні системи й електронні сховища документів.

– Інформаційно-довідкові системи.

– Прикладні системи персонального використання.

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

Система криптографічного захисту 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.

Головним перевагою системи криптографічного захисту є те, що вона бере на себе складних і потребуючих певних знань і кваліфікації функції спілкування із засобами криптографічного захисту 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 криптографічного захисту досить вивчити представлений в даному магістерському проекті.

Система криптографічного захисту 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 криптографічного захисту 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. Перевагою системи криптографічного захисту є й те, що це – тиражуємий продукт українського розроблювача. Це означає облік особливостей українських криптопровайдеров і низьку вартість володіння. Для використання системи криптографічного захисту 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.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо порівняльний аналіз існуючих програмних продуктів в області захисту інформації методом шифрування/дешифрування.

Truecrypt

Одна з декількох безкоштовних програм для шифрування даних, при цьому перевершує багато інших платних програм такого типу по функціональності. Підтримується велика кількість найвідоміших алгоритмів шифрування, у виборі вам допоможе вбудований модуль Benchmark-тестування. Інтуїтивно зрозумілий інтерфейс, створення віртуального диска проходить у вигляді Wizard'a. Віртуальний диск, як і у всіх вищеописаних програмах, можна автоматично примонтувати диск при запуску системи. Розмір віртуального диска обмежений лише можливостями операційної системи й поточному вільному місці на диску.

Можливості програми:

1. Створення й робота із шифрованими дисками.
2. Велика кількість алгоритмів шифрування.
3. Вбудований Benchmark-Модуль.

Drivecrypt Plus Pack

Drivecrypt Plus Pack – це програма для шифрування будь-яких дисків (фізичних, логічних і навіть системних), що має можливість використовувати передзагрузочну автентифікацію. Drivecrypt Plus Pack підтримує 256-бітне кодування даних у реальному часі за допомогою алгоритму шифрування (AES 256) на рівні секторів.

Можливості програми:

1. 256-бітне шифрування за допомогою AES-алгоритму;

2. Робота в режимі FDE (full disk encryption – кодування цілого диска);

3. Автентифікація користувача в передзагрузочній фазі (PBA-Pre-boot Authentication);

4. Можливість дворівневої автентифікації за допомогою зовнішніх електронних пристроїв типу USB Tokens від Aladdin і Rainbow;

5. Захист пароля від виявлення.

Bestcrypt

Пакет утиліт для створення на жорсткому диску комп'ютера віртуального зашифрованого диска (контейнера) – одного або декількох. Із зашифрованим контейнером можна працювати як зі звичайним жорстким диском – розміщати на ньому файли й робити з ними будь-які операції, інсталювати програми й т.ін. Шифрування/розшифровка йде у фоновому режимі, так що користувач не зауважує різниці в роботі зі звичайним і зашифрованим диском, що, до речі, при необхідності одним кличем може перетворитися у звичайний, але що не читається, файл.

Є в цієї програми й така опція, як створення "схованого" контейнера, яку не можна визначити (побачити) ніякими засобами. Крім цього, в bestcrypt входить утиліта для шифрування своп-файлу (у свопі зберігаються тимчасові дані, у тому числі документи й паролі, так що його шифрування є зовсім незайвою опцією для захисту інформації), а також bcwipe – утиліта, що дозволяє видаляти файли без можливості їхнього відновлення.

PGP Personal Desktop

Великий пакет програм, що представляє собою систему захисту інформації на комп'ютері від несанкціонованого доступу.

Можливості програми:

1. Шифрувати й ставити цифровий підпис (і, звичайно, навпаки) в електронних листах і вкладеннях;

2. Шифрування ICQ-трафіку;

4. Створення шифрованих дисків, папок і файлів;

5. Безпечне видалення даних з диска.

Vdfcrypt

Vdfcrypt – Програма для створення віртуальних зашифрованих дисків, причому ці віртуальні диски можна використовувати точно так само, як і звичайні жорсткі диски – установлювати на них програми, записувати дані й т.д. Є можливість створення й підключення образів CDROM і дискет; шифрування допускається робити як за допомогою оригінальних алгоритмів, так і використовуючи вже відомі (підтримуються SSE1, Ghost28147, IDEA, Blowfish, DES, 3DES, Arcfour).

У випадку непрацездатності в winxp з домашньої сторінки можна скачати патч.

Arcrypt

Arcrypt – Arcrypt (Advanced Russian Crypt) – програма для захисту даних. Створює віртуальний диск, що існує тільки під час сеансу роботи із шифрованою інформацією (для шифрування використовується алгоритм RCR2). На час сеансу віртуальний диск має всі характеристики логічного диска – на нього можна записувати файли й папки, видаляти їх, модифікувати й т.п. Всі ресурси, які видаляються, ретельно затираються, що виключає можливість їхнього відновлення якою-небудь утилітою. "Диск" закривається паролем, і цих паролів може бути декілька (кожному паролю буде відповідати свій зашифрований "диск"). Інтерфейс – російський, за допомогою скінов можна легко змінити зовнішній вигляд програми. Незважаючи на безкоштовність, програма вимагає реєстрації, що відбувається через Інтернет при її першому запуску.

Remora USB Disk Guard

Програма для шифрування даних на USB-носіях.

Шифрує файли й папки. Використовується 128-бітне шифрування.

Інтерфейс виконаний у вигляді USB-диска із кнопками "шифрувати обрані файли", "розшифрувати обрані файли", "шифрувати обрану папку", "розшифрувати обрану папку". На сайті розроблювача є також версія Pro з розширеними можливостями.

Cryptcd Enterprise Edition Retail

Давно відома й украй зручна програма, – проста, як все геніальне. Виготовляє зашифрований CD/DVD диск, що само запускається, із запитом пароля, тобто при запуску на будь-якому комп'ютері запросить пароль, а при перегляді його в провіднику виглядає як галявина однакових папок з дивними іменами. Використовує алгоритми шифрування: XOR, PGP, Flow Fish, 2 Fish, Scramble і AES. Тепер немає необхідності переживати, що залишили диск зі своїми даними на роботі, у друзів або у вас, його просто украли, відкрити або зрозуміти, що на ньому записано можна тільки при знанні пароля.

Filecoder

Filecoder – це надійна програма, що дозволяє шифрувати/дешифрувати як окремі файли, так і цілі групи файлів. Для шифрування використовується сучасна модифікація алгоритму DES – одного із кращих криптографічних алгоритмів, прийнятого як стандарт у США. Filecoder проста у використанні й не жадає від Вас яких-небудь знань в області криптографії. Програма працює під керуванням операційних систем сімейства Windows.

Dekart Private Disk Lite

Dekart Private Disk Lite – програма призначена для того, щоб сховати від сторонніх очей інформацію, яку їм краще не бачити, за допомогою створення віртуального зашифрованого диска будь-якого розміру. При настільки малому розмірі програми, вона дає захист, що обійти буде практично неможливо. При запуску ви побачите вікно у якому всього п’ять кнопок:

– Mount – нажавши на цю кнопку ви зможете підключити вже створений віртуальний диск;

– Create – нажавши на цю кнопку ви зможете створити віртуальний зашифрований диск;

– Delete – нажавши на цю кнопку ви зможете видалити раніше створений віртуальний диск, природно при цьому всі дані на ньому втратяться;

– Help – коротка довідка про функціональність кнопок і можливі дії;

– Dismount – кнопка, що дозволяє відключити підключений віртуальний диск, кнопка з'являється тільки тоді коли диск уже підключений.

Dekart Private Disk Lite

Ще одна програма із серії шифрувальників. Цього разу створювати диски не прийдеться, але й шифрувати можна тільки один файл, хоча ніхто не заважає вам запакувати кілька файлів в архів, а потім зашифрувати. Програма використовує відносно надійний алгоритм шифрування A 01-1225.

Процес шифрування файлу назвати складна мова не повернеться, вона, як і все геніальне, простий: Натискаємо кнопку "Зашифрувати", указуємо шлях до файлу, що збираємося закодувати й указуємо пароль, у відмінності від Dekart Private Disk Lite, пароль може бути довжиною до 1024 байт, але не менш двох символів. Після цих не складних операцій ви побачите, що з'явився файл із тією же назвою, що й шифруємий, але ще з додаванням розширення *.agf. Різниця в розмірах файлів усього в 6 байт.

Процес дешифрування так само простий. Натискаємо кнопку "Розшифрувати", указуємо шлях до файлу й вводимо пароль.

Знищення файлу відбувається в такий спосіб: натискаємо кнопку "Знищити" і вказуємо шлях до файлу, або видаляємо звичайними засобами Windows.

Також Agent 1024 уміє переглядати паролі, що перебувають в cache Windows.

Crypt4Free

Дозволяє зашифровувати будь-які файли на будь-якому носії, використовуючи один із двох наявних у програмі алгоритмів – Blowfish (448 бітний ключ) і DESX (128 бітний ключ).

Перед тим, як зашифрувати файл, потрібно буде встановити пароль, причому його можна згенерувати автоматично або набрати самому на віртуальній клавіатурі (робиться це для захисту від кейлоггерів), після чого нажати кнопку "Start".

Підтримується одночасне із шифруванням упакування файлів в zip-архіви; можливе швидке відправлення таких архівів по e-mail. Крім цього, можлива спільна робота з поштовою програмою для відправлення текстових повідомлень у зашифрованому виді.

Cryptoexpert

Програма для створення віртуальних зашифрованих дисків, які можуть підключатися як звичайні знімні диски.

Принцип роботи полягає в тому, що програма створює файл-контейнер, що представляє собою віртуальний зашифрований диск, з яким можна працювати, як із самим звичайним жорстким диском, тому що шифрування/розшифрування відбувається "на льоту". Підтримується криптостійке шифрування (BLOWFISH, CAST, 3DES), створення шифрованих дисків, доступних для роботи з локальної мережі, робота з USB-накопичувачами й багато чого іншого.

Lockdisk

Програма для шифрування даних. Створює зашифровані диски-контейнери, у яких можна розміщати будь-які файли. Обмеження безкоштовної версії: можливість створення дисків розміром не більше 35 MB. Крім обмеження на максимальний розмір створюваного диска, серед інших мінусів програми можна також відзначити той факт, що в документації зазначено, що lockdisk

використовує "256-бітне шифрування", але при цьому незрозуміло, який саме алгоритм застосовується.

The Bat! Private Disk

Програма для захисту даних – створює зашифрований файл-диск (один або трохи, при цьому використовується криптостійкий алгоритм AES), на якому можна зберігати конфіденційну інформацію. Процес шифрування/розшифрування відбувається в реальному масштабі часу, що робить його тим самим практично непомітним для користувача. Можливо мережне використання захищених даних, їхній запис на CD, FDD і будь-які інші носії інформації.

Max File Encryption

Це потужна й зручна програма для шифрування й захисту файлів. Програма Max File Encryption дозволяє зашифрувати файли в один EXE файл, доступ до якого можна одержати тільки знаючи пароль. Ви можете переглядати файли на будь-якому комп'ютері, при цьому не потрібне наявність настановних файлів програми Max File Encryption!

Також, програма має унікальну можливість: маскування файлів у звичайні файли (mp3, wav, bmp...), при якому вихідний файл залишається повністю працездатним!

Armor Tools

Набір інструментів, що забезпечує захист інформації на комп'ютері: Armor Tools дозволить працювати на комп'ютері, не залишаючи слідів своєї діяльності, видаляти документи й файли без можливості наступного відновлення, а також містить набір додаткових настроювань, що підвищують безпеку й продуктивність системи.

Kruptos

Програма для шифрування файлів і папок з використанням алгоритму Blowfish (128 біт). Має опцію невідновного видалення файлів і папок. Також дозволяє створювати зашифровані файли, що саморозпаковуються.

Схованка

Програма Схованка призначена для безпечного зберігання будь-якої персональної інформації користувача – телефони, поштові адреси, закладки, паролі й т.п. Має зручний інтерфейс і дуже проста в експлуатації. Можливості:

1. Зберігання персональної інформації будь-якого типу.
2. Простий і зрозумілий інтерфейс.
3. Гнучка база даних, що дозволяє створювати розділи під будь-які дані.
4. Груповий запуск посилань у розділі.
5. Швидкий пошук по розділах.
6. Пароль при запуску.
7. Шифрування бази.

Encrynote 0

Encrynote – простий шифрований блокнот на основі вкладок із самими основними функціями. Ідеально підходить для зберігання особистих записів. Основні функції:

– Шифрація тексту по паролю.

– Пошук по тексту в динамічному режимі, функція переходу до уведеного рядка.

– Мультимовність.

Sentry

Програма для шифрування даних в "прозорому" режимі – у масштабі реального часу. Використовуючи стійкі до злому алгоритми, створює зашифрований файл-контейнер, у якому й зберігаються ті дані, які потрібно сховати від сторонніх очей. Підтримується робота з даними, розташованими як на локальних дисках, так і на мережних, а також на змінних носіях, включаючи флеш-карти. Для спрощення використання Sentry 2020/XP інтегрується в Windows Explorer; підтримується керування гарячими клавішами, а також робота з командного рядка.

Storagecrypt

Storagecrypt дозволяє зашифрувати і закрити паролем інформацію, що перебуває на будь-яких змінних носіях – флеш-картах і т.п. Для шифрування застосовується криптостійкий алгоритм AES з 128-бітним ключем. Підтримується робота з файловими системами Fat12, Fat16, Fat32, NTFS.

Steganos locknote

Невелика й дуже зручна утиліта, призначена для шифрування заміток, у яких можна зберігати будь-яку приватну інформацію – паролі, серійники й будь-які інші невеликі за об'ємом дані (можна й великі, але "замітка" – Note – має на увазі короткий запис). У принципі, таких зашифрованих заміток може бути скільки завгодно: кожна з них після збереження зробленого запису буде виглядати виконується файлом, що (*.exe) із привласненим йому ім'ям – наприклад, справи.exe. Відкрити цей файл (тобто Прочитати або відредагувати запис) можна буде тільки після уведення пароля, що був зазначений перед збереженням цієї замітки.

Підтримується drag and drop; шифрування виробляється 256-бітним ключем AES, що забезпечує високу стійкість до злому.

Працює Steganos locknote без інсталяції.

Vipnet Safe Disk

Програма vipnet Safe Disk призначена для організації безпечного зберігання конфіденційної інформації на комп'ютері користувача. У процесі роботи програма створює захищені файли – контейнери. Файли – контейнери представляються користувачеві у вигляді логічних дисків операційної системи Windows. Наприклад, якщо в користувача на комп'ютері були диску A: C: D:, те при роботі програми vipnet Safe Disk будуть створюватися диски E: F: і т.д.

Без реєстрації працює 20 днів.

Ashampoo Magic Security

Програма для захисту конфіденційних даних будь-якого типу – файлів, особистих листів, приватних картинок, службових даних і т.п., що перебувають

на жорсткому диску, CD або DVD. Інтегрується з Windows Explorer, так що процес шифровки/розшифровки (підтримується стійкий до злому алгоритм AES) може виробляється дуже швидко й легко – в один клич.

Додаткові можливості: гарантоване – без можливості відновлення – видалення даних, шифрування даних одночасно з їхнім записом на CD або DVD, інтеграція з поштовим клієнтом, створення зашифрованих архівів, що самораспаковуються.

Інтерфейс – багатомовний, оцінний період роботи – 30 днів. Для використання потрібно одержати безкоштовний trial-ключ.

Powerkey

Програма для шифрування інформації, доступ до якої потрібно обмежити. Підтримується можливість шифрування даних одним з восьми алгоритмів шифрування, а також стиск даних в ZIP архів. Крім цього, за допомогою powerkey можна створювати SFX архіви, що самораспаковуються, а також безповоротно видаляти непотрібні дані (файли й папки). Можлива інтеграція програми в оболонку Windows.

Strongdisk

Strongdisk Pro – комплексна криптографічна система для захисту конфіденційної інформації при зберіганні на персональних комп'ютерах, ноутбуках і змінних носіях. Strongdisk Pro створює в системі Windows захищені диски, доступ до яких надається по паролі й/або ключу. Уміст захищених дисків шифрується на льоту. Є набір засобів для запобігання витоку інформації "повз захищені диски". Деякі інші можливості: приховання самого факту наявності на комп'ютері конфіденційної інформації; миттєве блокування доступу до інформації з натискання «гарячої клавіші» або при добуванні електронного ключа; миттєва підмінена щирої інформації на заздалегідь підготовлену помилкову в критичній ситуації. .

Обмеження незареєстрованної версії – дозволяє працювати з "дисками" розміром не більше 3 мб.

Evidence Eliminator

Evidence Eliminator – захисний інструмент, що дозволяє швидко і якісно видалити інформацію, що бачити стороннім ні до чого. Це може бути й інформацією про сайти, які ви відвідали – навіщо шефові/дружині/дітям (непотрібне викресліть) знать, куди ви заглядаєте, і розкидані по всьому диску логи від додатків, і та інформація, що "вилучена" з кошика, а насправді може бути дуже легко відновлена, і файл підкачування, з якого теж можна витягти вкрай цікаву (для чужих очей) інформацію, і ... – не буду перераховувати всі, на що здатно програму. Її остання версія має 69 модулів, кожний з яких відповідає за доручений йому ділянка роботи – видалити певну інформацію про вашу діяльність, як би далеко й глибоко вона не була захована. І всі вони зі своїм завданням справляються на відмінно.

Без реєстрації програма проробить 30 днів.

Scramdisk

Програма створює віртуальний шифрований диск, доступ до якого закритий паролем. Увівши пароль, з віртуальним диском можна працювати як зі звичайним, навіть інсталювати програми. Для шифрування можна використовувати різні алгоритми (Triple-DES, IDEA, MISTY1, Blowfish, TEA, Square).

Крім властиво шифрування, scramdisk уміє ще й ховати зашифровану інформацію у вже наявні файли формату *.wav – сторонній не зрозуміє, що в музиці зберігається важлива інформація.

INFO Hider

Утиліта для шифрування даних, що використовує стеганографію – приховання самого факту наявності зашифрованої інформації. Для цього INFO Hider ховає приховувану інформацію усередині графічного файлу, що зовні практично не міняється, а лише трохи збільшується в розмірі (у порівнянні з оригіналом).

Stealth Encryptor

Програма для шифрування даних (підтримує стандарти DES і Blowfish). Дозволяє не тільки зашифровувати дані, що перебувають на жорсткому диску, але приховувати їх від сторонніх очей, створювати архіви, що само розпаковуються, які можна переносити на інші комп'ютери (не мають цієї програми), а також використовувати зашифровану e-mail-переписку й приховання "чутливих" даних у самих звичайних графічних файлах (картинках). Підтримується інтеграція із Провідником (робота з контекстного меню).

Windefender

За допомогою цієї програми можна зберігати коштовні дані в зашифрованому виді: при роботі з ними (після введення пароля) windefender буде автоматично розшифровувати їх при зчитуванні з диска й зашифровувати назад при їхньому записі на диск (для шифрування файлів використовується криптостійкий, стійкий до злому симетричний алгоритм шифрування ДЕРЖСТАНДАРТ 28147-89). Шифрування/розшифрування при цьому буде відбуватися абсолютно непомітно – windefender використовує спеціальний низькорівневий драйвер, що з рівним успіхом шифрує як документи, так і файли, що виконуються.

Крім основної функції – шифрування даних, програма володіє й іншими коштовними властивостями – приміром, вона дозволяє встановлювати на окремі папки й файли права доступу – наприклад, дозволити тільки читання файлів з певної папки, або взагалі неї сховати. Ще одна корисна, але не відразу помітна особливість цієї програми – допустимість зберігання зашифрованих даних на компакт-диску (windefender буде розшифровувати файли прямо з CD).

Без реєстрації windefender дозволяє захистити тільки один каталог або файл.

Signature Cryptographer

Програма для захисту інформації у важливих файлах від несанкціонованого доступу. Захист здійснюється шляхом шифрування даних

файлу. Шифрувальник використовує як ключ уміст файлів замість рядка пароля. Таким чином, довжина пароля може досягати гігантських розмірів (або зовсім бути більше довжини шифруємого файлу, що робить зашифрований файл теоретично не зламуваним).

Замість довгих рядків пароля запам'ятати потрібно тільки ім'я файлу, використовуваного для пароля. Спосіб шифрування файлів заснований на алгоритмі шифрування "Сигнатура", докладно описаному в інтерактивній довідковій системі.

Finecrypt

Створює зашифровані файли, що самовитягаються, які можна розшифровувати й на тих комп'ютерах, де finecrypt не встановлений. Головне – знати пароль або мати ключ...

Програма дозволяє використовувати один з десяти алгоритмів шифрування, підтримує Drag & Drop, інтегрується в оболонку Windows; є в неї й така корисна опція, як безповоротне видалення даних.

Безкоштовний варіант дозволяє працювати не більш ніж з п'ятьома файлами; у випадку покупки програми ($49) у ній з'являються значно більше можливостей, включаючи шифрування архівів.

Safedisk

Система захисту інформації. Дозволяє створювати й використовувати віртуальні шифровані диски. Підтримує алгоритми RIJNDAEL, BLOWFISH, SQUARE, GOST, SKIPJACK, RSC. Невидимий режим роботи (stealth). Вбудований захист від програм – клавіатурних шпигунів. Режим роботи "під примусом". Блокування доступу до комп'ютера при завантаженні/простої. Низькорівневий wipe." (wipe – це в цьому випадку видалення інформації без можливості її відновлення якими-небудь утилітами).

Програма може запускатися автоматично із завантаженням Windows і працювати при цьому в невидимому режимі, що дозволяє сховати сам факт присутності захисту на комп'ютері.

У випадку виникнення надзвичайної ситуації можна скористатися режимом аварійного закриття диска (нажавши певну комбінацію клавіш); крім цього, є режим "робота під примусом", що дозволяє знищити інформацію при уведенні спеціального пароля.

Без реєстрації програма проробить 30 днів.

ICE Encrypt

"Проста програма шифровки файлів/каталогів. ICE Encrypt використовує симетричний блоковий алгоритм нового покоління – icesafe. Icesafe перевершує ВСІ ВІДОМІ ОПУБЛІКОВАНІ симетричні блокові алгоритми, такі як (DES, AES, blowfish, GOST, ...). Icesafe не просто черговий алгоритм шифрації, це нова ідеологія захисту інформації."

Mooseoft Encrypter

Програма для шифрування файлів зі зручним інтерфейсом і можливістю працювати через контекстне меню. Допускає вибрати один з алгоритмів шифрування: Blowfish, Cast128, Cast256, GOST, IDEA, MARS, Misty1, RC2, RC5, RC6, Rijndael або Twofish. Уміє створювати зашифровані файли, що самовитягають, які можна, наприклад, відправити по e-mail, не турбуючись про те, що дані можуть бути перехоплені. Крім цього, програма дозволяє очищати жорсткий диск, знищуючи файли без можливості їхнього відновлення, а також уміє автоматично створювати безпечні паролі.

Safepassword

Компактна база даних високої надійності для зберігання паролів – safepassword задіє ряд найбільш стійким, перевіреним часом і практикою криптографічних алгоритмів з метою забезпечити високу надійність довгострокового зберігання паролів, імен користувачів і інших подібних даних у файлах на дисках і інших носіях інформації.

Іспитовий період: 100 запусків програми без реєстрації, без нагадувань і інших перешкод. Зареєстровані користувачі програми можуть користуватися safepassword безкоштовно.

Система захисту інтернет трафіку

Система захисту трафіку – програма для туннелінгу TCP-з'єднань по HTTP протоколі. Аналог VPN. Проходить через більшість файрволів і проксі. Дуже корисна для офісних працівників, яким системні адміністратори закривають порти. Працює на комп'ютері користувача як локальний транслятор портів + socks4 proxy + http proxy. Таким чином, якщо в користувача закритий ICQ або IRC, те він може використовувати програму як проксі і весь трафік пройде по HTTP тунелю. Крім того, трафік пройде зашифрованим і неможливо буде нічого бачити за допомогою пакетного фільтра – сніффера. Сисадмину не видно куди заходив користувач і чого робив. Корисна звичайним домашнім користувачам для захисту від сніффингу з боку провайдерів і від інших ловців паролів. Буде корисна для використання в публічних Wi-Fi мережах. Трафік іде зашифрований до сервера в США й далі до пункту призначення як звичайно. Програма корисна в якості анонімайзера; крім того, на льоту стискає текстовий трафік (html, txt) і, таким чином, створює економію при оплаті трафіку."

Securelock

Ховає, а при бажанні й шифрує файли, папки й диски (FAT12,16,32) шляхом відповідної модифікації файлової системи, причому заховані файли виявляться невидимими для програм типу Disk Editor; не видні вони й у Безпечному й DOS-режимі. Для роботи потрібно Microsoft .NET Framework.

Без реєстрації ховається до 140 МБ даних.

S-Tools

Програма для приховання даних від сторонніх очей. Працює без інсталяції. Використовує метод стеганографії – дані ховаються у файлах графіки (BMP або GIF) або музичному WAV-файлі, які зовні не відрізняються від аналогічних файлів, що не несуть захованої інформації (картинку можна подивитися, музику можна послухати). Справа в тому, що оцифровані файли (ті ж *.bmp або *.wav) можуть бути деякою мірою змінені, і це не вплине на якість звуку або зображення (вірніше, ці зміни будуть практично не помітні). Крім

цього, програма дозволяє не тільки сховати інформацію в зовні нічим не примітному файлі, але й зашифрувати її. Звичайно, у порівнянні з лідером серед подібних програм – програмою Steganos Security Suite, S-Tools виглядають більш слабо – відсутні такі приємні можливості, як приховання даних не тільки в аудіофайлах і картинках, але й в HTML або текстовому файлі, немає функції видалення файлів без можливості їхнього відновлення.

Animabilis RS File Encryption

Animabilis RS File Encryption – програма шифрування файлів на знімних носіях. У наш час величезне поширення одержали різні пристрої змінної пам'яті, такі як флеш-драйви, карти пам'яті, знімні вінчестери. Animabilis RS File Encryption – легка у використанні, надійний і потужний засіб для захисту важливої інформації, яка зберігається на змінних носіях. Установивши програму на ваш пристрій ви можете використовувати її на будь-якому комп'ютері де встановлена Windows.

Animabilis RS File Encryption дозволяє шифрувати файли будь-якого типу, включаючи документи Microsoft Word, Excel і powerpoint, і запобігає несанкціонованому перегляду і зміні інформації. У випадку втрати або пропажі змінного носія, ваші дані залишаться закритими для сторонніх. Для шифрування застосовується надійний алгоритм Blowfish. Паролі шифрування не зберігаються усередині зашифрованого файлу. Даний фактор дозволяє одержати дійсно стійке шифрування.

Програма має інтуїтивно зрозумілий і зручний інтерфейс, будь-який користувач навіть що не має пізнань в області захисту інформації зможе без праці захистити свої дані. У програму вбудована можливість створення зашифрованих файлів, що саморозпаковуються. Ви можете переглядати ваші зашифровані файли на будь-якому комп'ютері, при цьому не потрібне наявність настановних файлів програми. Основні можливості:

1. Шифрування файлів за допомогою алгоритму Blowfish.

2. Дешифрування файлів.

3. Створення зашифрованих файлів, що саморозпаковуються.

4. Надійне видалення файлів.

5. Робота на будь-якому змінному носії (якщо пристрій підключений до ОС MS Windows).

6 Ведення логу файлу всіх операцій.

Dekart Secrets Keeper

Secrets Keeper захищає інформацію в середовищі Windows Explorer і Microsoft Office. Secrets Keeper дозволяє шифрувати (AES) файли будь-якого типу, включаючи документи Microsoft Word, Excel і powerpoint, і запобігає несанкціонований перегляд і зміні інформації. Доступ до зашифрованої інформації може бути здійснений тільки при пред'явленні правильного пароля або PIN-коду, якщо використовується USB-ключ або смарт-карта. Без пароля або PIN-коду зашифрований файл зовсім не читаємий.

Masker

Програма для захисту даних, що використовує стеганографію – приховання зашифрованих даних у зовні необразливих файлах – графічних (bmp, gif, jpg, tif), музичних (wav, mid, snd, mp3), відео (avi, mov, mpg) і навіть в *.exe або *.dll, при цьому файли, у яких "ховаються" зашифровані відомості, залишаються повністю функціональними.

FET XP

FET XP створений для захисту інформації, яка зберігається, на комп'ютері від несанкціонованого доступу засобами криптографії. Так само в програму вбудована функція повного й безповоротного видалення файлів з жорсткого диска або інших носіїв що дозволяють перезаписувати інформацію (дискети, магнитооптичні диски, ZIP диски й т.п.). Алгоритм шифрування із застосуванням відкритого ключа, дозволяє обмінюватися інформацією з мережі Internet, не побоюючись її перехоплення, і внесення змін. Дана криптосистема побудована на основі сучасних відкритих криптоалгортмів (подробиці в технічному описі),

стійкість* яким обґрунтована математичними доказами й доведена сучасними криптографами.

Shella

Shella – легка у використанні, й надійний засіб шифрування конфіденційної інформації (файлів, тексту) і надійного видалення файлів з дисків. Shella дозволяє шифрувати файли будь-якого типу, у тому числі й документи Microsoft Office, запобігаючи їхню несанкціоновану зміну або перегляд. Для шифрування застосовується криптостійкий алгоритм AES (Rijndael) з довжиною ключа 256 біт. Функція гешування SHA2 дозволяє перетворити звичайний пароль у криптостійкий ключ. На відміну від декількох подібних програм пароль шифрування не зберігається разом з файлом, що дозволяє забезпечити дійсно надійний захист. Програма проста в обігу й не вимагає особливих навичок роботи за комп'ютером, тому освоїти її зможе будь-який користувач.

Shella невимоглива до ресурсів і буде працювати на будь-якому комп'ютері із установленою системою Windows. Без реєстрації: 50 запусків.

Cryptoheaven

Багатофункціональна система засекречування, що включає в себе просту й зашифровану e-mail переписку, зашифровану "аську", а також допускає безпечне спільне використання ресурсів і не менш безпечну передачу файлів. Допускається багатокористувальницьке використання.

Трохи бентежить те, що, незважаючи на високий рівень захисту даних, обмін інформацією відбувається через центральний сервер, а це, у принципі, дозволяє контролювати весь, нехай і зашифрований (ключами володіє тільки відправник і одержувач), трафік.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне

забезпечення, яке призначено для системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис технологій центру шифрування

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.

Симетрична криптографія

Основні поняття

Розглянемо загальну схему симетричної, або традиційної, криптографії.

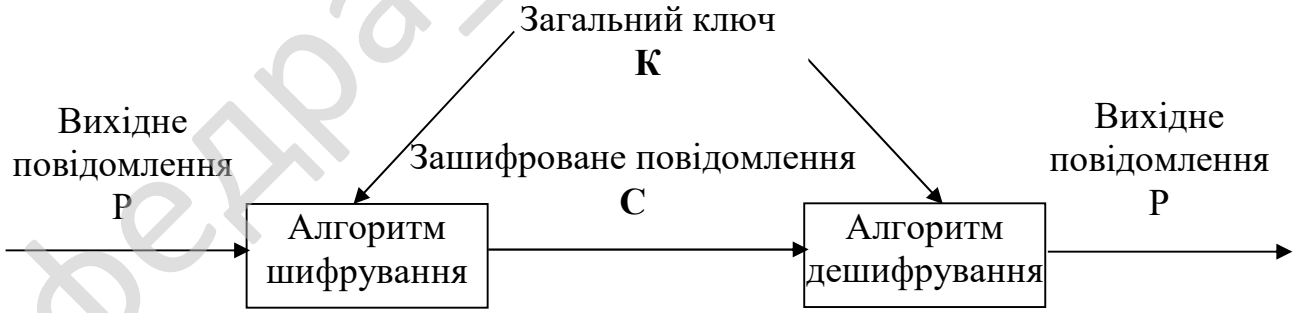

Рисунок 3.1 – Загальна схема симетричного шифрування

У процесі шифрування використовується певний алгоритм шифрування, на вхід якому подаються вихідне незашифроване повідомлення, називане також

plaintext, і ключ. Виходом алгоритму є зашифроване повідомлення, називане також ciphertext. Ключ є значенням, що не залежить від шифруємого повідомлення. Зміна ключа повинна приводити до зміни зашифрованого повідомлення.

Зашифроване повідомлення передається одержувачеві. Одержувач перетворить зашифроване повідомлення у вихідне незашифроване повідомлення за допомогою алгоритму дешифрування й того ж самого ключа, що використовувався при шифруванні, або ключа, легко одержуваного із ключа шифрування.

Незашифроване повідомлення будемо позначати P або M, від слів plaintext і message. Зашифроване повідомлення будемо позначати C, від слова ciphertext.

Області застосування

Стандартний алгоритм шифрування повинен бути застосуємо в багатьох додатках:

– Шифрування даних. Алгоритм повинен бути ефективний при шифруванні файлів даних або великого потоку даних.

– Створення випадкових чисел. Алгоритм повинен бути ефективний при створенні певної кількості випадкових біт.

– Гешування. Алгоритм повинен ефективно перетворюватися в однобічну геш-функцію.

Платформи

Стандартний алгоритм шифрування повинен бути реалізований на різних платформах, які, відповідно, висувають різні вимоги.

– Алгоритм повинен ефективно реалізовуватися на спеціалізованій апаратурі, призначеної для виконання шифрування/дешифрування.

– Великі процесори. Хоча для найбільш швидких додатків завжди використовується спеціальна апаратура, програмні реалізації застосовуються частіше. Алгоритм повинен допускати ефективну програмну реалізацію на 32-бітних процесорах.

– Процесори середнього розміру. Алгоритм повинен працювати на мікроконтролерах і інших процесорах середнього розміру.

– Малі процесори. Повинна існувати можливість реалізації алгоритму на смарт-картах, нехай навіть із урахуванням твердих обмежень на пам'ять.

Додаткові вимоги

Алгоритм шифрування повинен, по можливості, задовольняти деяким додатковим вимогам.

– Алгоритм повинен бути простим для написання коду, щоб мінімізувати ймовірність програмних помилок.

– Алгоритм повинен мати плоский простір ключів і допускати будь-який випадковий рядок біт потрібної довжини як можливий ключ. Наявність слабких ключів небажано.

– Алгоритм повинен легко модифікуватися для різних рівнів безпеки й задовольняти як мінімальним, так і максимальним вимогам.

– Всі операції з даними повинні здійснюватися над блоками, кратними байту або 32-бітному слову.

Мережа Фейштеля

Блоковий алгоритм перетворює n-бітний блок незашифрованого тексту в n-бітний блок зашифрованого тексту. Число блоків довжини n дорівнює 2^n. Для того щоб перетворення було оборотним, кожний з таких блоків повинен перетворюватися у свій унікальний блок зашифрованого тексту. При маленькій довжині блоку така підстановка погано приховує статистичні особливості незашифрованого тексту. Якщо блок має довжину 64 біта, то він уже добре приховує статистичні особливості вихідного тексту. Але в цьому випадку перетворення тексту не може бути довільним у силу того, що ключем буде саме перетворення, що виключає ефективну як програмну, так і апаратну реалізації.

Найбільш широке поширення одержали **мережі Фейштеля**, тому що, з одного боку, вони задовольняють всім вимогам до алгоритмів симетричного шифрування, а з іншого боку, досить прості й компактні.

Мережа Фейштеля має наступну структуру. Вхідний блок ділиться на трохи рівної довжини підблоків, називаних гілками. У випадку, якщо блок має довжину 64 біта, використовуються дві гілки по 32 біта кожна. Кожна гілка обробляється незалежно від інший, послу чого здійснюється циклічне зрушення всіх гілок уліво. Таке перетворення виконується кілька циклів або раундів. У випадку двох гілок кожний раунд має структуру, показану на рисунку:

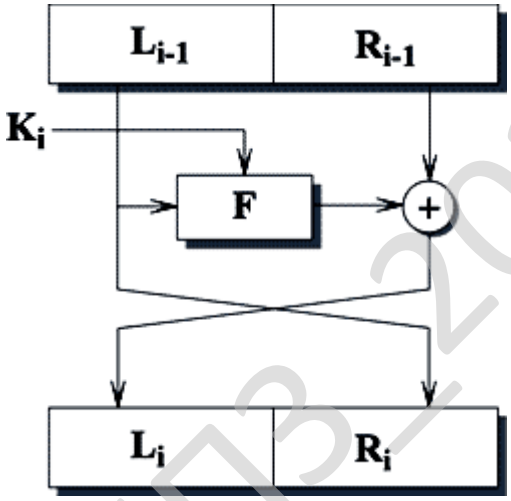

Рисунок 3.2 – і-ий раунд мережі Фейштеля

Функція `F` називається утворюючою. Кожний раунд складається з обчислення функції `F` для однієї гілки й побітового виконання операції `XOR` результату `F` з іншою гілкою. Після цього гілки міняються місцями. Вважається, що оптимальне число раундів – від 8 до 32. Важливо те, що збільшення кількості раундів значно збільшує криптостійкість алгоритму. Можливо, ця особливість і вплинула на настільки активне поширення мережі Фейштеля, тому що для більшої криптостійкости досить просто збільшити кількість раундів, не змінюючи сам алгоритм. Останнім часом кількість раундів не фіксується, а лише вказуються припустимі межі.

Мережа Фейштеля є оборотною навіть у тому випадку, якщо функція `F` не є такою, тому що для дешифрування не потрібно обчислювати F^{-1}. Для

дешифрування використовується той же алгоритм, але на вхід подається зашифрований текст, і ключі використовуються у зворотному порядку.

Останнім часом все частіше використовуються різні різновиди мережі Фейштеля для 128-бітного блоку із чотирма гілками. Збільшення кількості галузей, а не розмірності кожної гілки пов'язане з тим, що найбільш популярними дотепер залишаються процесори з 32-розрядними словами, отже, оперувати 32-розрядними словами ефективніше, ніж з 64-розрядними.

Основною характеристикою алгоритму, побудованого на основі мережі Фейштеля, є функція `F`. Різні варіанти стосуються також початкового й кінцевого перетворень. Подібні перетворення, називані забілюванням (whitening), здійснюються для того, щоб виконати початкову рандомизацію вхідного тексту.

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 3.3 – 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.

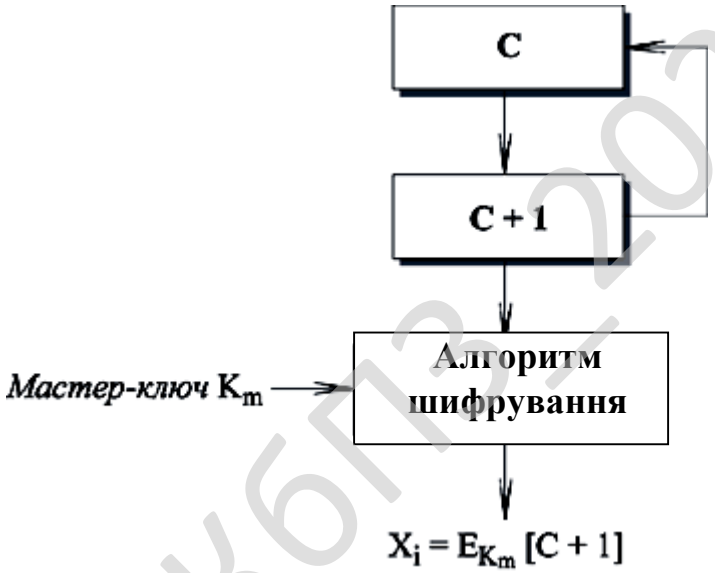

Рисунок 3.4 – Циклічне шифрування

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 – і-ий біт геш-кода, $1 \leq i \leq n$.

– k – число n-бітних блоків входу.

– b_{ij} – і-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.

3.2 Розробка структурної схеми

На рисунку 3.5 зображено структурну схему розробленого програмного забезпечення.

Програмне забезпечення структурно складається з наступних блоків:

– Головний модуль програми.

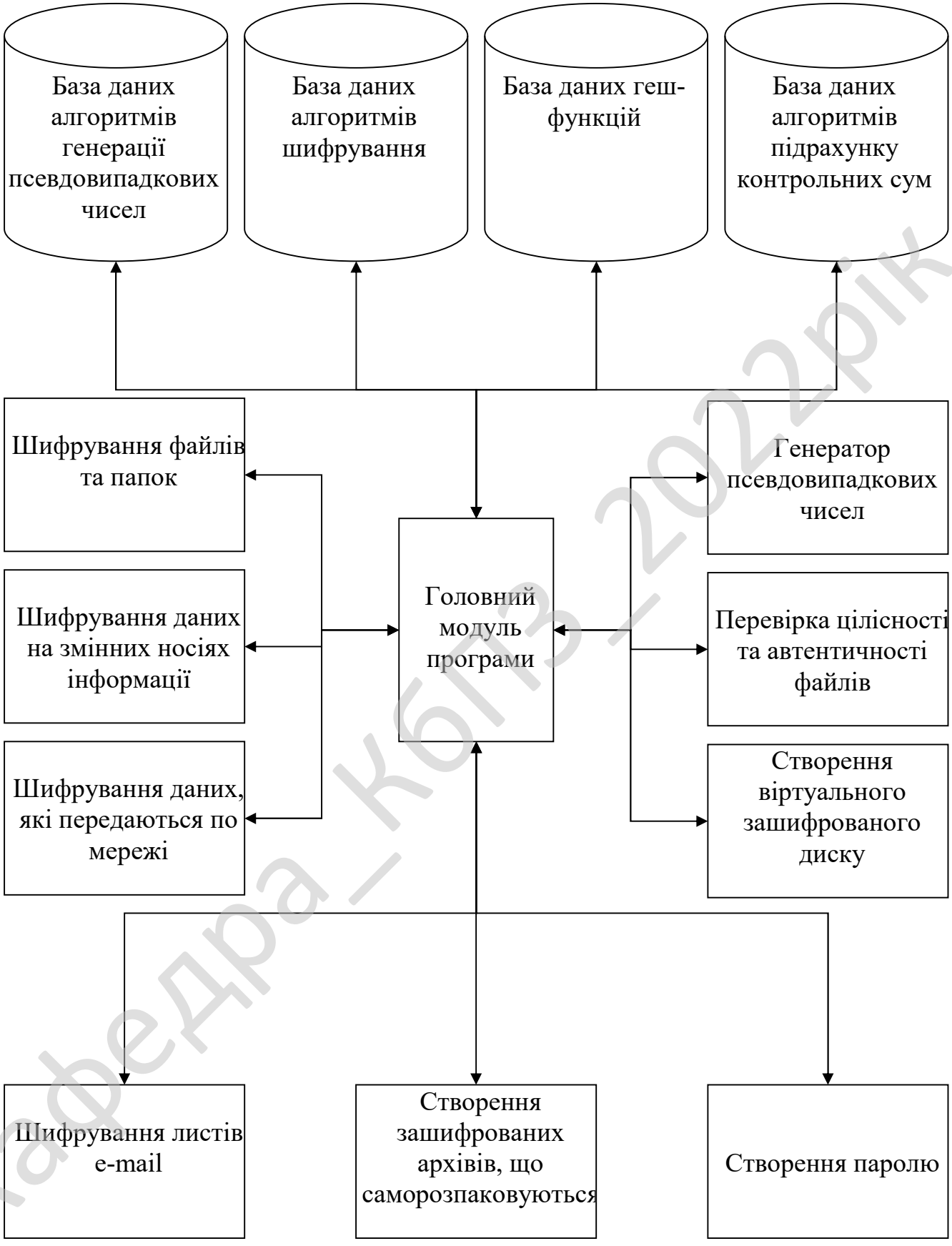

Рисунок 3.5 – Структурна схема розробленого програмного забезпечення

– База даних алгоритмів шифрування.

– База даних геш-функцій.

– База даних алгоритмів генерації псевдовипадкових чисел (ГПВЧ).

– База даних алгоритмів підрахунку контрольних сум (CRC).

– 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.

3.3 Розробка функціональної схеми

На рисунку 3.6 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

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

– Блок алгоритмів шифрування.

– Блок геш-функцій.

– Блок алгоритмів генерації псевдовипадкових чисел (ГПВЧ).

– Блок алгоритмів підрахунку контрольних сум (CRC).

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.

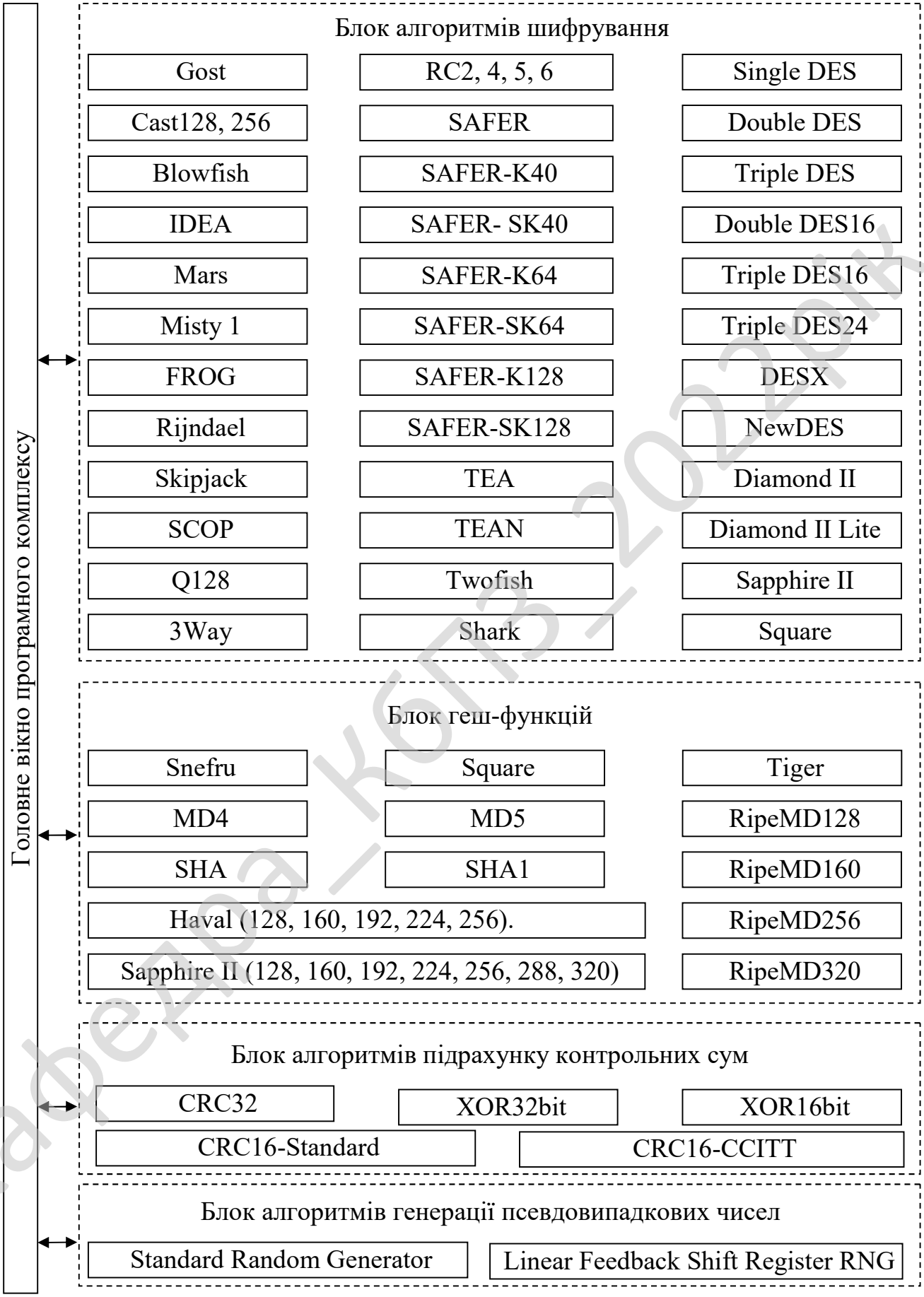

Рисунок 3.6 – Функціональна схема системи

У блоці алгоритмів підрахунку контрольних сум реалізовані наступні алгоритми:

– CRC32.

– XOR32bit.

– XOR16bit.

– CRC16-CCITT.

– CRC16-Standard.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.7 .

З нього ми бачимо, що у системі відбуваються наступні процеси.

Спершу завантажується процес початку/кінця програми, який взаємодіє з наступними процесами:

– Процес вибору файлу.

– Процес вибору режиму шифрування.

Процес вибору файлу взаємодіє з процесом вибору геш-функції, який у свою чергу взаємодіє з наступними процесами:

– Обчислення геш-функції.

– Вибір алгоритму шифрування.

Процес вибір алгоритму шифрування дозволяє вибрати алгоритм, за допомогою якого буде зашифрований файл, або інший носій даних.

Він взаємодіє з наступними процесами:

– Введення ключа шифрування.

– Вибір режиму шифрування.

Процес Вибору режиму шифрування, є одним з головних процесів, які

відбуваються при завантаженні розробленого, у результаті виконання магістерського проектування, програмного забезпечення.

Він взаємодіє з наступними процесами:

– Вибір алгоритму шифрування.

– Процес шифрування.

– Процес дешифрування.

– Процес початку/кінця роботи.

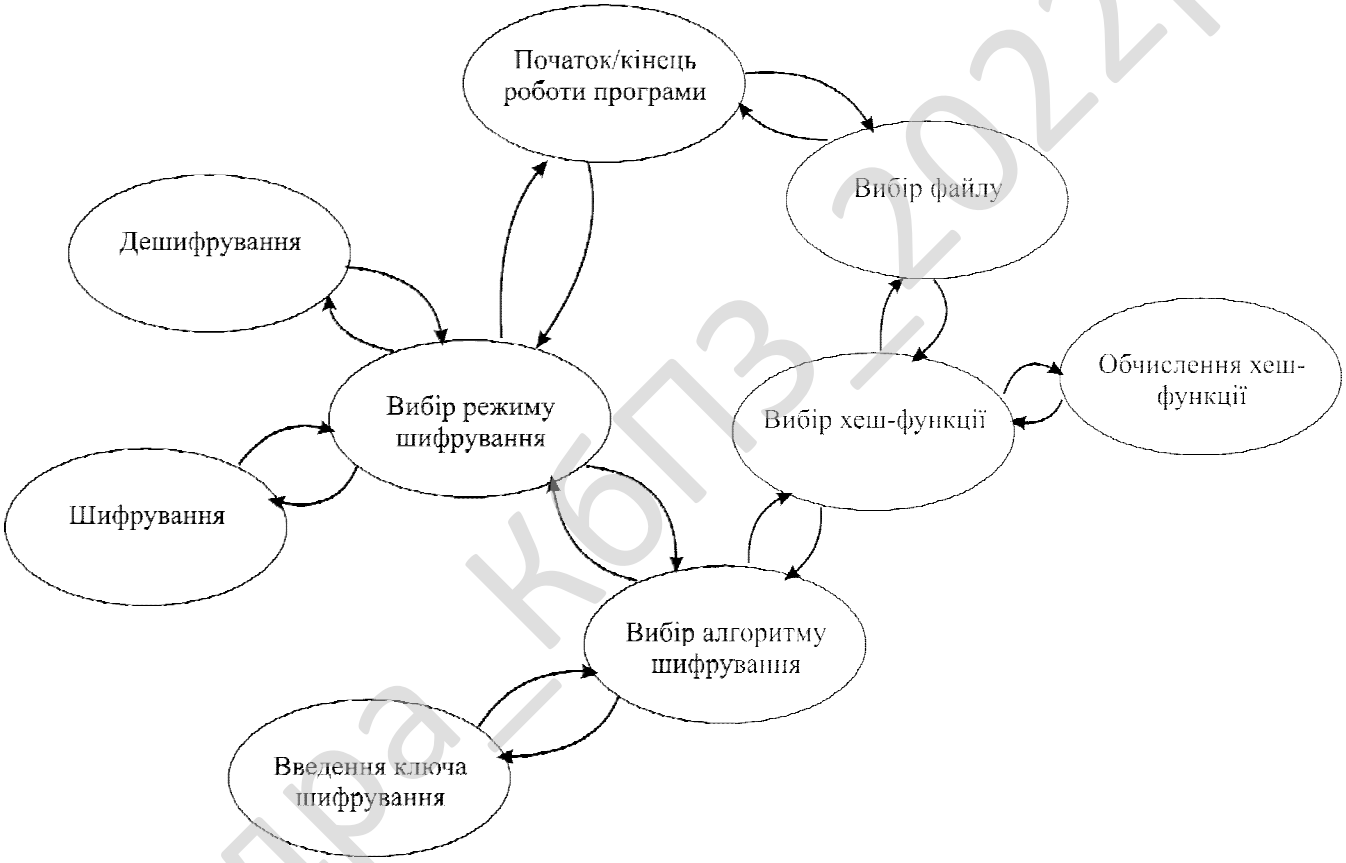

Рисунок 3.7 – Діаграма процесів системи

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

Спершу відбувається виведення основного вікна програми. Після цього відбувається введення шляху до файлу, який потрібно зашифрувати.

Користувачеві надається можливість створити геш-функцію, для файлу.

Якщо він не хоче створювати геш-функцію, то відбувається перехід до блоку шифрування.

У іншому випадку, тобто, якщо потрібно створити геш-функцію, то відбуваються наступні кроки:

– Вибір геш-функції.

– Обчислення геш-функції.

– Виведення результату обчислень.

Після цього відбувається перехід до блоку шифрування або дешифрування.

Блок шифрування складається з наступних кроків:

– Виклик функції вибору алгоритму шифрування.

– Введення ключа.

– Шифрування файлу.

Блок дешифрування складається з наступних кроків:

– Виклик функції вибору алгоритму шифрування.

– Введення ключа.

– Дешифрування файлу.

Після виконання усіх вище перерахованих дій, програма завершує свою роботу.

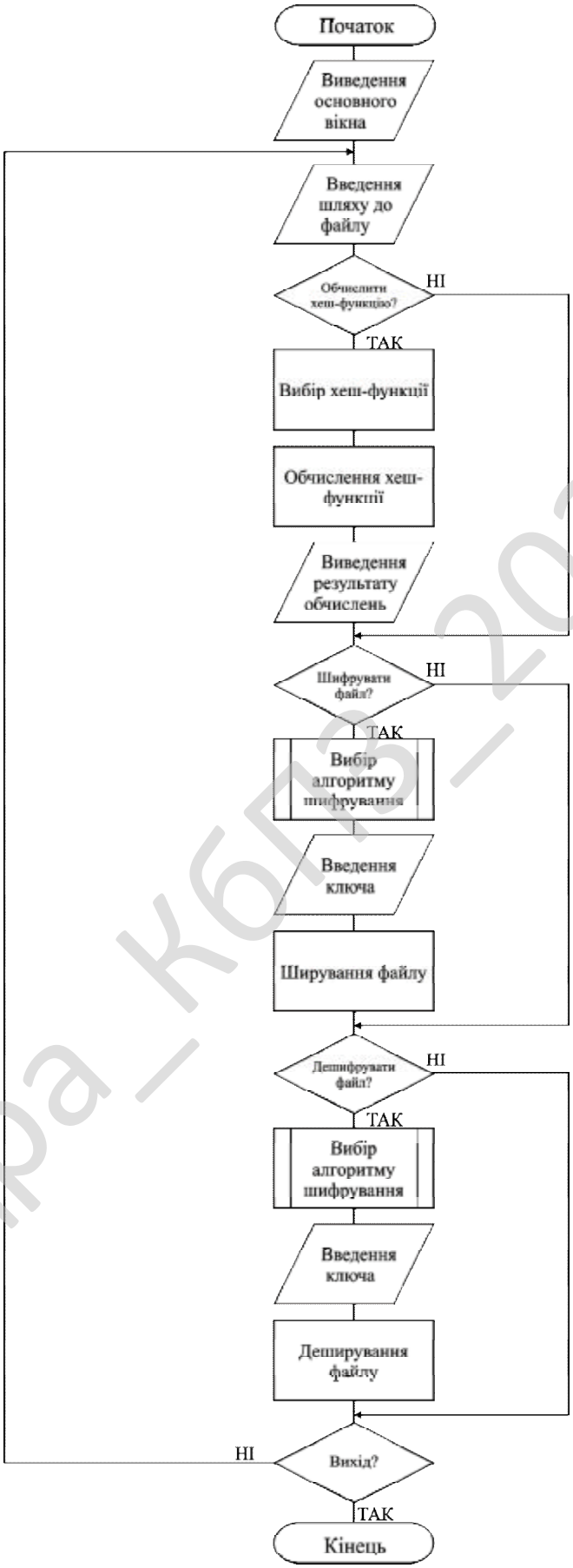

Рисунок 4.1 – Блок-схема роботи основної програми

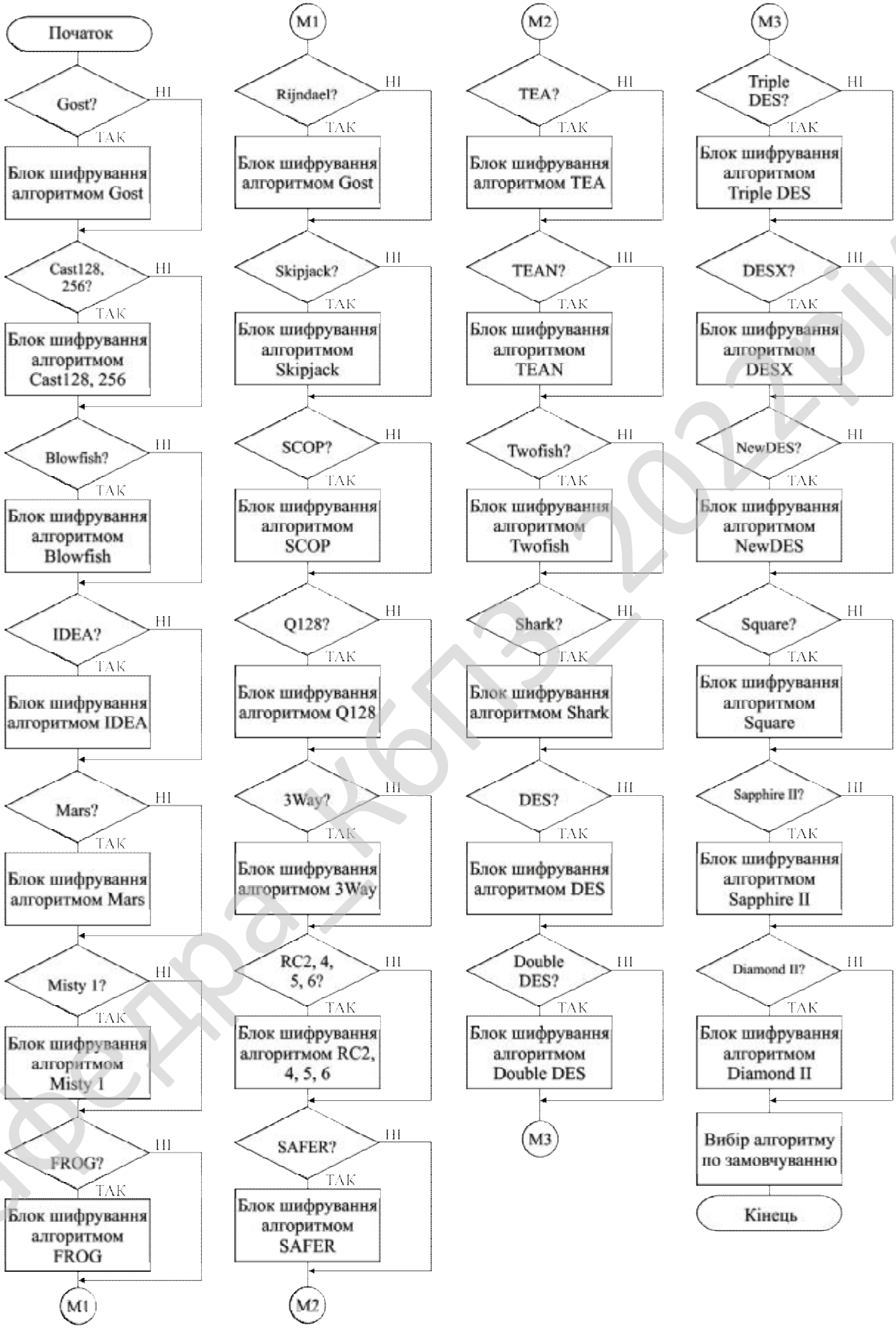

Рисунок 4.2 – Блок-схема роботи підпрограми алгоритму шифрування

На рисунку 4.2 зображено блок-схему підпрограми алгоритму шифрування.

Алгоритм складається у виборі одного з методів шифрування та виконання криптографічних перетворень, використовуючи цей метод, над даними, які потрібно зашифрувати, або розшифрувати.

Підпрограма дозволяє вибрати один з наступних методів шифрування:

- 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.

Нижче приведемо програмний код, який дозволяє зареєструвати ці класи алгоритмів шифрування.

```
{$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 розширений', '');
  RegisterCipher(TCipher_Twofish, '', '');
{$ENDIF}
```

Нижче приведемо код опису основних алгоритмів.

```
// 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;
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 28147:2009, що є класичним алгоритмом симетричного шифрування на основі мережі Фейстеля (рисунок 4.3). Даний алгоритм шифрує інформацію блоками по 64 біта (такі алгоритми називаються "блоковими"). Зміст мережі Фейстеля полягає в тому, що блок шифруемої інформації розбивається на два або більше субблоків, частина яких обробляється за певним законом, після чого результат цієї обробки накладається (операцією побітового додавання за модулем 2) на необроблювані субблоки. Потім субблоки міняються місцями, після чого обробляються знову й т.д. певне для кожного алгоритму число раз – раундів.

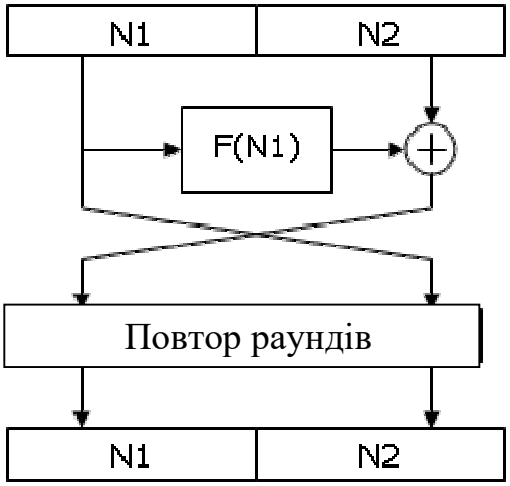

Рисунок 4.3 – Мережа Фейстеля

Основна відмінність алгоритмів симетричного шифрування друг від друга складається саме в різних функціях обробки субблоків. Дана функція часто називається "основним криптографічним перетворенням", оскільки саме вона несе основне навантаження при шифруванні інформації. Основне перетворення алгоритму ДСТУ 28147:2009 є досить простим, що забезпечує високу швидкодію алгоритму; у ньому виконуються наступні операції (рисунок 4.4).

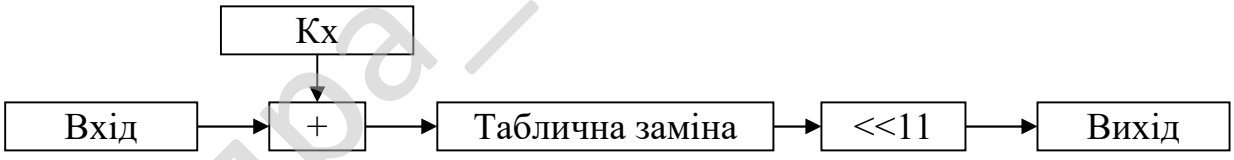

Рисунок 4.4 – Основне перетворення алгоритму ДСТ 28147:2009

1. Додавання субблоку з певним фрагменом ключа шифрування за модулем 2^{32}. Кх – це 32-бітна частина ("підключ") 256-бітного ключа шифрування, якому можна представити як конкатенацію 8 підключів: К = $К_0К_1К_2К_3К_4К_5К_6$ $К_7$. Залежно від номера раунду й режиму роботи алгоритму (про їх – нижче), для даної операції вибирається один з підключів.

2. Таблична заміна. Для її виконання субблок розбивається на 8 4-бітних фрагментів, кожний з яких прогоняється через свою таблицю заміни. Таблиця заміни містить у певній послідовності значення від 0 до 15 (тобто всі варіанти значень 4-бітні фрагменти даних); на вхід таблиці подається блок даних, числове подання якого визначає номер вихідного значення. Наприклад, подається значення 5 на вхід наступної таблиці: "13 0 11 74 91 10 143 5 122 15 8 6". У результаті на виході виходить значення 9 (оскільки 0 заміняється на 13, 1 – на 0, 2 – на 11 і т.д.).

3. Побітове циклічне зрушення даних усередині субблока на 11 біт уліво.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмний продукт розроблений в результаті виконання магістерського проектування, є дуже зручним та прости у використанні.

Інтерфейс користувача наведений на рисунках 5.1 – 5.5.

На рисунку 5.1 наведене головне вікно програми, з якого ми бачимо, що користувач може обрати або функцію гешування, або функцію шифрування інформації.

На головній формі, вказуються наступні дані:

– 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.

– Геш значення файлу.

– Зашифрований геш.

– Дешифрований геш.

На рисунку 5.2 зображене випадаюче меню вибору геш-функції, у якому перелічені усі геш-функції, які ми можемо використати у програмі.

На рисунку 5.3 зображене меню вибору алгоритму шифрування, у якому перелічені усі алгоритми шифрування, які ми можемо використати у програмі.

На рисунку 5.4 зображене меню вибору режиму шифрування, у якому перелічені усі режими шифрування, які ми можемо використати у програмі.

На рисунку 5.5 зображена інформація про розробника.

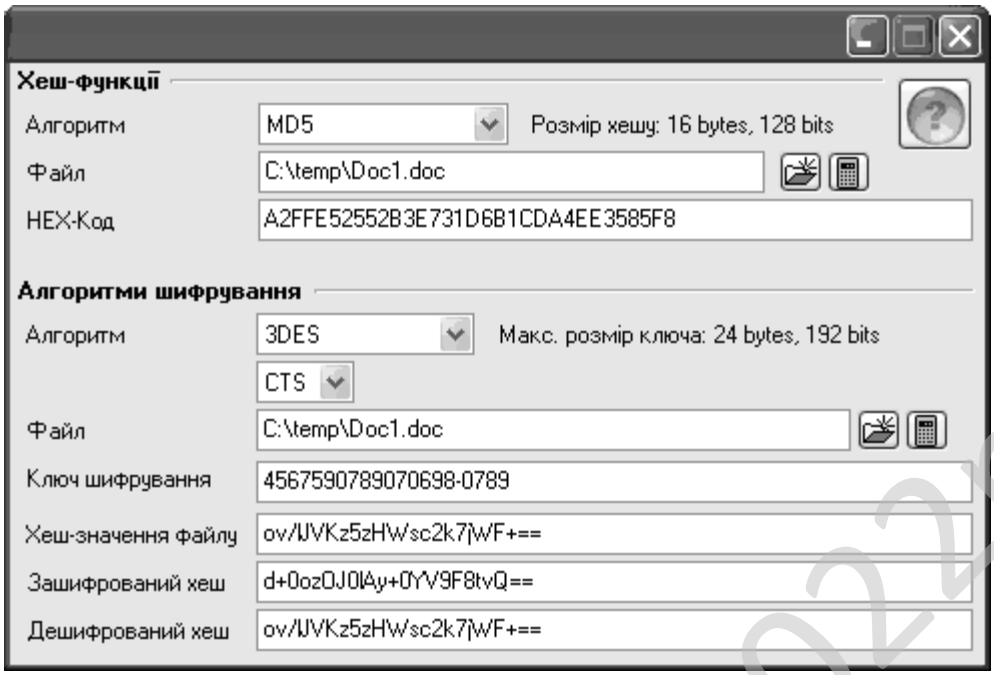

Рисунок 5.1 – Головне вікно програми

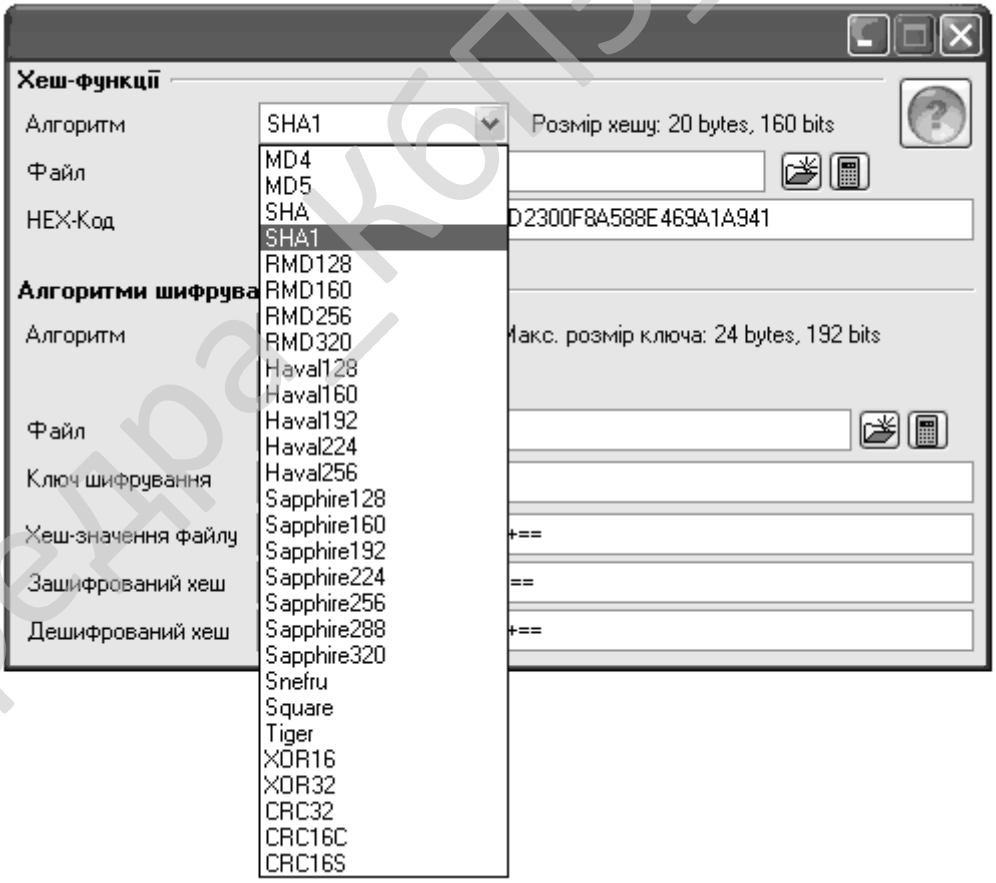

Рисунок 5.2 – Вибір геш-функції

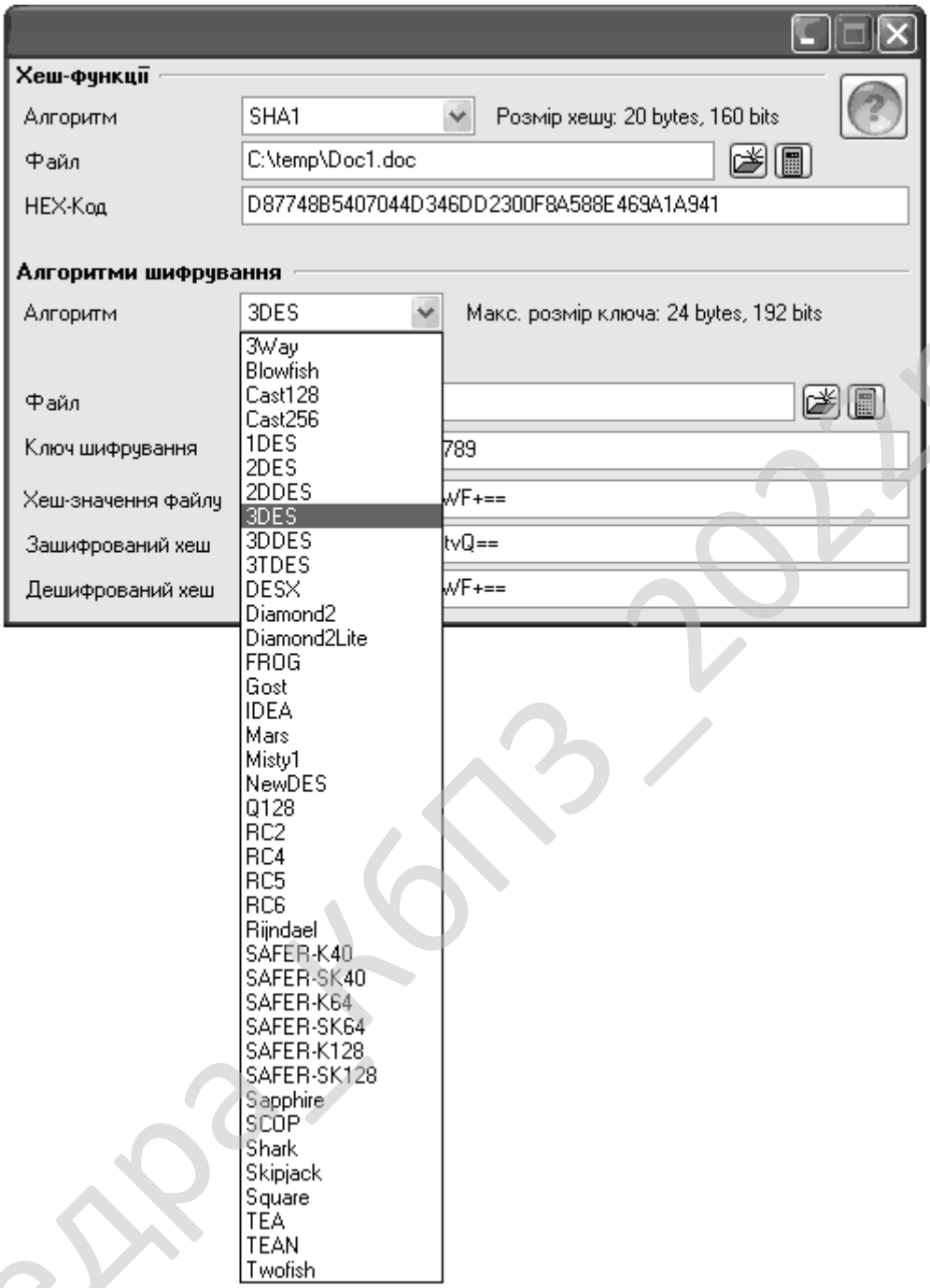

Рисунок 5.3 – Вибір алгоритму шифрування

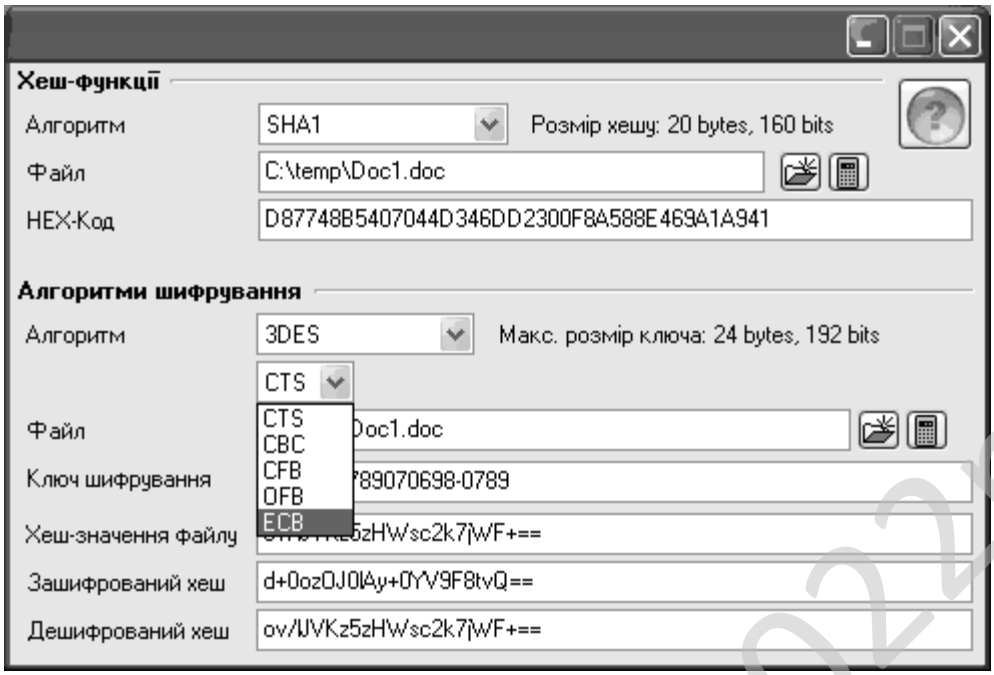

Рисунок 5.4 – Вибір режиму шифрування

Рисунок 5.5 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Метою розробки є дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Об'єктом дослідження є процес з використанням мультиваріантного центру реалізації криптоалгоритмів.

Предметом дослідження є методи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

– Удосконалено метод з використанням мультиваріантного центру реалізації криптоалгоритмів.

– Розроблено вітчизняний продукт з використанням мультиваріантного центру реалізації криптоалгоритмів, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

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

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

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи для протидії декомпіляції коду.

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

а) невеликий розмір;

б) невеликі системні потреби;

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

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	50
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

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

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	50000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

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

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

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

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

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

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

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

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

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

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

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

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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

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

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

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot3{,}23\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot65=133 \text{ люд/день}$$

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

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	133	Ф 7.1-7.4
Впровадження	13	Д13
Всього	174	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{174 \cdot 1}{60\text{-}5} = 3,2 \text{ ставки}$$

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

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

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

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	6	540	9
Монітор	60	6	360	6
Клавіатура	30	6	180	3
Маніпулятор «мишка»	30	6	180	3
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	35,99

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{36 \cdot 3}{1,2} = 129,6 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=129,6/(60·8)=0,25 ставки

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

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

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

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

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	0,25	12000	9000
Продакт-менеджер	0,25	8640	6480
Інженер-програміст	3,2	10300	98880
Інженер-електронщик	0,25	8000	6000
Інженер-системотехнік	0,25	8000	6000
Адміністратор мережі	0,5	8000	12000
Системний програміст	0,25	8000	6000
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	8000	6000
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$=5,95	-	$Ф_{роб}$=171360

Розрахуємо середньоденну зарплату одного виконавця:

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

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{171360}{5,95 \cdot 60} = 480 \ грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

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

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

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sy – питома площа на одне робоче місце, $м^2$; $Ц_{пл}$ – вартість одного квадратного метра площі, грн.

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

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

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

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

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

де Цм – ціна меблів для одного робочого місця, грн.

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за пропозицією інтернет ресурсу hotline за 16.10.22 – джерело https://hotline.ua

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок Dell OptiPlex 7040 Tower		7347
Процесор	Intel Core i5-6400 (4 ядра по 2.7 – 3.3 GHz), 6 MB Smart Cache	-
Системна плата	Dell OptiPlex 7040 6x USB 3.0, 4x USB 2.0, 1x HDMI, 2x DP, 2x Audio, 1x COM-порт, 1x LAN (RJ-45), 2x PS/2	-
Відеокарта	Інтегрована в процесор Intel HD Graphic 530 (до 1792 MB з ОЗП)	-
Жорсткий диск	SSD 120Gb + HDD Seagate Barracuda 750 Gb	-
Оперативна пам’ять	8Gb DDR4 non-Reg., no-ECC, (модулі 2x4Gb)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	Dell OptiPlex Tower, 500W (80mm fan), full-ATX, БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 3.0, 2xUSB 2.0, Audio, silver/black	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	50000	10	5000
Разом	$К_р$= 1606075		$А_р$= 141140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

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

де Ne – Кількість екземплярів програм, шт.

$$З_о = 480 \cdot 174/50 = 1672 \text{ грн}$$

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

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

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 1672 \cdot 10 \cdot 0{,}01 = 167{,}2 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

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

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(1672+167{,}2) = 681 \text{ грн}$$

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

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 1672 \cdot 15 \cdot 0{,}01 = 251 \text{ грн}$$

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

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

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

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

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

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

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д \,, \tag{7.17}$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 41,2 грн./шт.

$$З_{М2} = 41{,}2 \cdot 10 = 412 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з ., \tag{7.18}$$

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

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

$$З_М = (105 + 412 + 1702)/50=44 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_п = 1672 \cdot 15 \cdot 0{,}01=251 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 50 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 141140 \cdot 3/(50 \cdot 12)=706 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

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

$$C_п = 1672+167+681+251+44+251+706 = 3772 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п , \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 3772 = 2075 \text{ грн.}$$

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

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	1672
2. Додаткова зарплата виконавців	$З_д$	167
3. Відрахування на соціальні потреби	$С_{оц}$	681
4. Загальногосподарські витрати	$Г_{осп}$	251
5. Витрати на матеріали	$З_М$	44
6. Освоєння нових операційних систем, мов програмування	$О_п$	251
7. Амортизація основних фондів	$А_м$	706
8. Повна собівартість програмного забезпечення	$С_п$	3772
9. Плановий прибуток	$П_р$	2075
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	5847
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1169,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	7922

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

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

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

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	7922
Всього капітальних витрат	–	7922

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	42944	8052
2. Витрати на електроенергію	$З_{ел}$	2565	1282
3. Витрати на амортизацію	$З_{ам}$	0	1981
Всього витрат за рік	$І$	45509	11315

Витрати на оплату праці:

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

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин обслуговування системи зменшилась з 400 годин до 75 годин на рік, тому витрати на обслуговування склали:

$$З_{р\ баз} = 400 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 42944\ грн.$$

до

$$З_{р\ нов} = 75 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 8052\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

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

Зел баз=·0,475·2455·2,2 = 2565 грн

Зел нов=·0,475·1227·2,2 = 1282 грн

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	7922	–	1980,5
Всього відрахувань	-	–	7922	–	1980,5

7.8 Визначення економічної ефективності програмної продукції

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

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

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

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

$$Ев = (5847 - 3772) \cdot 50 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 50000) \cdot 3/12 = 68465 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

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

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

$$T_в = \frac{198075}{(5847 - 3772) \cdot 50 \cdot 12 / 3} = 0,5 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (45509 - 11315) - 0,25 \cdot 7922 = 32214 \text{ грн.}$$

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

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

$$T_{сп} = \frac{7922}{45509 - 11315} = 0,2 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	50
2. Повна собівартість розробленої програми	Грн.	3772
3. Ціна розробленої програми	Грн.	5847
4. Плановий прибуток від реалізації розробленої програми	Грн.	2075
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1606075
7. Загальний прибуток від реалізації програмної продукції	Грн.	103750
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	68465
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	7922
11.Величина економічного ефекту у користувача програмної продукції	Грн.	32214
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

8.2 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому

покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об'єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

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

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

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

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

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

викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

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

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

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств, центрів комп’ютерної техніки, центрів інформаційних технологій, сучасних комп’ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров’я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Для більшого розуміння, пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значенн;

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

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [9].

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

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

8.5 Розрахункова частина

Для захисного штучного заземлення будемо застосовувати вертикальні електроди з сталевого прокату круглого перерізу діаметром 35 мм., довжиною *L=2 м.,* та горизонтальний електрод – металева полоса з перетином 35·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру (прямокутником).

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,6 м*. Відстань між вертикальними заземлювачами (електродами) *A=3 м*. Глибина закладення горизонтального контура заземлення *t=0,75 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,75+2/2=1,75 \text{ м.}$$

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

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

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому грунті [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Діаметр вертикального електрода (задан):

$$Dв = 35 \text{ мм.} = 0,035 \text{ м.}$$

Відношення *A/L=3/2=1,5.*

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

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

$$=0,366(54,5/2)[(lg(2\cdot2/0,035)+(1/2\ )lg((4\cdot1,75+2)/(4\cdot1,75-2))]=$$

$$=21,7\ Ом.$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев=0,53$ при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [11].

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

$$N=R_O\ /(Кев\ R_{ЗН})=\ 21,7\ /\ (0,53\cdot4)=\ 10,2\approx 10\ шт.$$

Визначаємо довжину з'єднуючої полоси:

$$L_П\ =1,05\cdot A\cdot N=\ 1,05\cdot3\cdot10=32,3\approx 32\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

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

$$=0,366(40\cdot5/40)\cdot lg((2\cdot40^2)/(0,035\cdot0,75\ ))=11,14\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної кліматичної зони для з'єднуючої полоси [11]:

$B = 35\ мм. = 0,035\ м.$- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

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

$$=(21,7\cdot\ 11,14)/(21,7\cdot\ 0,55+\ 10\cdot\ 11,14\cdot\ 0,53)=3,4\ Ом.$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

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

Так як R суттєво більше $R_{ЗН}$, зменшимо кількість вертикальних електродів до *9* і виконаємо перерахунок. У результаті остаточно отримали: кількість вертикальних електродів дорівнює *9* при $R = 3,9\ Ом.$

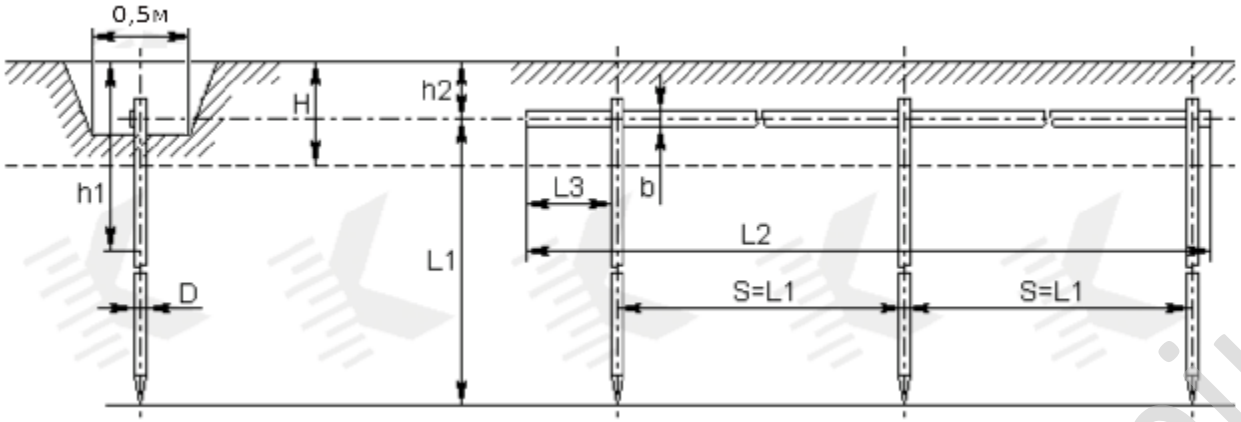

Рисунок 8.1 – Схема штучного заземлення

8.6 Висновки до розділу

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

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

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

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

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів з використанням мультиваріантного центру реалізації криптоалгоритмів.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем з використанням мультиваріантного центру реалізації криптоалгоритмів.

– Досліджена система з використанням мультиваріантного центру реалізації криптоалгоритмів.

– На основі отриманих результатів досліджень створена програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання з використанням мультиваріантного центру реалізації криптоалгоритмів.

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

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

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

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 28147:2009.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 32214 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фадєєв М.О. Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

3. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

4. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

5. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

6. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

7. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

8. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

9. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

10. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

11. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

12. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

13. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов,

А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. – практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

14. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

15. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

16. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

17. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

18. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

19. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

20. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

21. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

22. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

23. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

24. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

25. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов,

С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

26. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

27. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

28. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

29. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

30. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

31. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

32. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

33. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

34. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

35. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

36. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов,

С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня – 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

37. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

38. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

39. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез ІІІ міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

40. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая – 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

41. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. – техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

42. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

43. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

44. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

45. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

46. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

47. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

48. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

49. A.B. Bagula, M. Botha, and A.E Krzesinski. Online Traffic Engineering: The Least Interference Optimization Algorithm // IEEE Communications Society – 2004, P. 1232-1236.

50. Anees Shaikh, Jennifer Rexford, and Kang G. Shin. Evaluating the Impact of Stale Link State on Quality-of-Service Routing // IEEE/ACM Transactions on Networking. – 2001. – №9(2), P. 162-176.

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

52. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-

007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

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

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

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

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

57. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0024.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Фадєєв М.О.			Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов С.А.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

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

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи з використанням мультиваріантного центру реалізації криптоалгоритмів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

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

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

8 Вимоги щодо охорони праці

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

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 117 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

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

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 25.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов С.А.

Дослідження та програмна реалізація системи з використанням мультиваріантного центру реалізації криптоалгоритмів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 65

Літера: РП

Кропивницький – 2022 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:='МАГІСТЕРСЬКА РОБОТА';
Label2.Caption:='на тему:';
Label3.Caption:='Дослідження та програмна реалізація системи з використанням
мультиваріантного центру реалізації криптоалгоритмів';
Label4.Caption:='';
Label5.Caption:='Керівник: Смірнов С.А.';
Label6.Caption:='Розробив: студент Фадєєв Максим Олексійович ';
Label7.Caption:='гр. КІ-21М-1,4';
Label8.Caption:='м. Кропивницький 2022';
end;

end.
```