МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

С. П. Євсеєв
О. В. Мілов
О. Г. Король

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ОСНОВ КРИПТОГРАФІЧНОГО ЗАХИСТУ

Навчальний посібник

Харків
ХНЕУ ім. С. Кузнеця
2020

УДК 004.056.55(076)
Є25

Авторський колектив: д-р техн. наук, доцент С. П. Євсеєв – вступ, лабораторні роботи 1, 5; канд. техн. наук, доцент О. В. Мілов – лабораторні роботи 2, 3, 7; канд. техн. наук, доцент О. Г. Король – лабораторні роботи 4, 6, додатки.

Рецензенти: професор кафедри захисту інформації Національного університету "Львівська політехніка", д-р техн. наук, доцент *І. Р. Опірський*; завідувач кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету, д-р техн. наук, професор *О. А. Смірнов*.

Рекомендовано до видання рішенням ученої ради Харківського національного економічного університету імені Семена Кузнеця.
Протокол № 5 від 26.12.2019 р.

Самостійне електронне текстове мережеве видання

Євсеєв С. П.

Є25 Лабораторний практикум з основ криптографічного захисту [Електронний ресурс] : навчальний посібник / С. П. Євсеєв, О. В. Мілов, О. Г. Король. – Харків : ХНЕУ ім. С. Кузнеця, 2020. – 222 с.
ISBN 978-966-676-791-5

Подано лабораторні роботи, що призначені для практичного вивчення питань використання методів захисту в інформаційних системах. Розглянуто основи послуги безпеки, методи їхньої реалізації на основі симетричних та асиметричних алгоритмів шифрування. Досліджено алгоритми цифрового підпису. Запропоновано практичні основи створення захищеного середовища з використанням програмного комплексу PGP, вивчено принципи побудови стеганографічних систем, а також проведення статистичних досліджень генераторів випадкових та псевдовипадкових послідовностей за методикою NIST STS.

Рекомендовано для студентів, які навчаються за напрямами "Кібербезпека", "Програмна інженерія", "Інформатика" і "Комп'ютерні науки" всіх форм навчання, та інших спеціальностей, де вивчають цикл навчальних дисциплін із захисту інформації, а також для самостійного опанування його основ.

УДК 004.056.55(076)

ISBN 978-966-676-791-5

Вступ

Стрімке зростання глобальної мережі "Інтернет" і розвиток інформаційних технологій привели до формування інформаційного середовища, яке безпосередньо впливає на всі сфери людської діяльності. Нові технологічні можливості значно полегшують обмін інформацією, її пошук та зберігання. Нормою сьогодні є спілкування за допомогою мережі, електронний документообіг і залучення "хмарних" ресурсів для різного роду інформаційних потреб підприємств та окремих користувачів.

У зв'язку із цим загострилися питання інформаційної безпеки, як корпоративної, так і персональної. Під інформаційною безпекою сьогодні розуміють захищеність конфіденційності, цілісності та доступності інформації, а також підтримувальної інфраструктури, від випадкових і навмисних впливів, які можуть призвести не тільки до моральних або фінансових збитків, але, скажімо, у разі бізнес-структур, й до повного краху підприємства.

Звісно, у таких умовах зростає потреба у фахівцях у галузі розроблення програмного забезпечення, які володіють основними аспектами захисту інформації, та здатні виконати розроблення інформаційної системи із забезпеченням необхідних послуг безпеки, стійку до відомих загроз, яка має адекватний рівень захисту.

Саме тому метою навчальної дисципліни є засвоєння необхідних знань з основ інформаційної безпеки, а також формування твердих практичних навичок та набуття компетентностей щодо використання сучасних механізмів захисту інформації.

Формуванню таких навичок, на думку авторів, і буде сприяти запропонований цикл лабораторних робіт, завдання у якому підібрано таким чином, що дозволяють студенту поступово опанувати важливі питання захисту інформації. Поступове ускладнення понятійного апарату дасть змогу краще зрозуміти питання, пов'язані з механізмами інформаційної безпеки. Дослідження державних і сучасних міжнародних стандартів буде сприяти набуттю відповідних компетентностей із розроблення політики інформаційної безпеки.

Посібник містить опис семи лабораторних робіт. Кожна з них розкриває відповідну тему навчальної дисципліни, висвітлюючи практичні питання захисту інформації. Склад кожної лабораторної роботи містить такі складові частини:

мету лабораторної роботи;

рекомендації до підготовки до виконання роботи;

основні положення;

завдання до лабораторної роботи за варіантами;

хід виконання лабораторної роботи (у разі потреби);

контрольні запитання.

У ході проведення лабораторної роботи студент має продемонструвати:

знання понятійного апарату та основних криптографічних перетворень, що вивчають у роботі;

розуміння принципів функціонування та використання криптопротоколів;

грамотне використання програмного забезпечення, що реалізує відповідні методи захисту інформації;

навички в побудові елементів захисту інформаційної системи.

Вивчення тем та виконання завдань містить такі етапи:

1. Підготовчий етап (до проведення практичного або лабораторного заняття):

а) отримання відповідного цим методичним рекомендаціям завдання, номера варіанта та вимог викладача;

б) вивчення теоретичного матеріалу за відповідною темою;

в) розроблення алгоритму виконання завдання.

2. Безпосереднє виконання завдання в аудиторії, комп'ютерному класі обчислювального центру або самостійно (в іншому місці, наприклад, удома):

а) проходження допуску до лабораторної роботи;

б) установлення (за потреби) конфігурування програмного забезпечення;

в) відпрацьовування завдання за варіантом;

г) аналіз здобутих результатів.

3. Складання звіту і захист роботи.

Звіт про лабораторну роботу має містити:

титульний лист із назвою лабораторної роботи і даними виконавця;

дату виконання;

мету роботи;

опис завдання;

тексти розроблених HTML-документів і програм;

результати роботи та їхній аналіз;

висновки про роботу.

Усі матеріали звіту необхідно зброшурувати.

Лабораторна робота 1
Найпростіші шифри

1.1. Мета

Мета цієї лабораторної роботи – ознайомитися з основними принципами роботи найпростіших шифрів; вивчити елементарні криптографічні операції: перестановки та заміни (підстановки), проаналізувати їхні властивості; вивчити елементи криптоаналізу, зокрема частотного аналізу криптограм.

1.2. Рекомендації до підготовки до виконання

Для успішного виконання лабораторної роботи (ЛР) необхідно володіти математичними поняттями перестановки та заміни, принципами шифрування і розшифрування текстових повідомлень, а також розуміти модель криптоаналітика.

1.3. Загальні теоретичні положення ЛР

Повідомлення – це текст, малюнок, аудіозапис, зміст або інші змістовні електронні дані, що містять конфіденційну інформацію.

Відкритий (вихідний) текст – це повідомлення, яке необхідно зробити недоступним для сторонніх осіб, тобто змінити його змістовну логіку. Множину відкритих текстів позначають як *M*.

Зашифрований (закритий) текст – це повідомлення зі зміненим логічним змістом, яке також називають криптограмою або шифротекстом. Множину криптограм позначають як *C*.

Ключ – це секретне значення деякого параметра або параметрів, що забезпечує вибір одного криптографічного перетворення із сукупності можливих для використаного методу шифрування. Ключем може бути випадкове число або вектор. Множину ключів позначають як *K.*

Шифр – це сукупність однозначних зворотних перетворень множини можливих відкритих текстів на множину можливих шифротекстів, що здійснюються за певними правилами із застосуванням ключа.

Шифрування – це процес перетворення повідомлень, у результаті виконання яких відкриті повідомлення відображаються в шифротекст. Процес шифрування даних позначають як *E.*

Розшифрування – це вид перетворення повідомлення, що здійснюють, із метою відновлення змісту інформації, прихованої в зашифрованому тексті (шифротексті), у результаті виконання якого зашифроване повідомлення (шифротекст) відображається у відкрите повідомлення. Процес розшифрування даних позначають як *D.*

Криптоаналіз – це галузь криптології, яка розглядає проблеми зламування шифрів, із метою відновлення відкритої інформації або такої фальсифікації зашифрованої інформації, яка в результаті має бути прийнята за справжню.

Симетричні алгоритми розподіляють на дві категорії. Одні з них обробляють текст побітово (або побайтово) і називають потоковими алгоритмами або потоковими шифрами. Ті ж, які працюють із групами бітів відкритого тексту, називають блоковими алгоритмами (шифрами). У лабораторній роботі розглядають найпростіші симетричні шифри, які ґрунтуються на найпростіших арифметичних операціях підстановки, перестановки й елементах модулярної арифметики. Тому всі прості симетричні шифри можна розподілити на дві групи:

1) шифри перестановок;

2) шифри підстановок:

а) одноалфавітні;

б) поліалфавітні.

Алфавітом називають довільну скінчену впорядковану множину A, за допомогою якої записано повідомлення (відкриті тексти) (рис.1.1).

№	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A_y	А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М

№	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
A_y	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	_

Рис. 1.1. **Приклад українського алфавіту для запису відкритого тексту**

Перестановкою називають розташовані в певному порядку *k*-елементи упорядкованої множини *M* або алфавіту.

Підстановка – це взаємно однозначне відображення *f* деякої кінцевої множини *M* на множину *M'* або на себе. Водночас у підстановці беруть участь усі елементи множин *M* і *M'*.

Шифри перестановок. Перестановні шифри ґрунтуються на алгебраїчній операції, що називають перестановкою, унаслідок виконання якої здійснюють шифрування відкритого тексту та розшифрування криптограми.

Шифр перестановок із ключовим словом. Літери відкритого тексту записують у клітинки прямокутної таблиці у її рядках. Літери ключового слова пишуть над стовпчиками та вказують на їхній порядок (за зростанням номерів букв в алфавіті). Щоб отримати зашифрований текст, треба виписувати літери у стовпчиках з урахуванням їхнього алфавітного порядку (рис. 1.2).

Приклад:

Відкритий текст: захист інформації.

Ключ: шифр.

ш	и	ф	р
4	1	3	2
з	а	х	и
с	т		і
н	ф	о	р
м	а	ц	і
ї			

Рис. 1.2. **Отримання зашифрованого тексту за допомогою шифру перестановок із ключовим словом**

Криптограма: атфа_иірі_х_оц_зснмї.

Ключове слово (послідовність стовпців) відоме адресату, який легко зможе розшифрувати повідомлення.

Матрична перестановка з подвійним ключем. Більш складною матричною перестановкою є перестановка з подвійним ключем. Відкритий текст записують у матрицю за певним ключем $K_1 = \{1, 2, \ldots, d_1\}$, що залежить від довжини тексту. Криптограма утворюється під час зчитування із цієї матриці за ключем $K_2 = \{1, 2, \ldots, d_2\}$. Розмірність матриці дорівнює $d_1 \times d_2$.

Приклад:

Відкритий текст: "*ШИФРУВАННЯ_ПЕРЕСТАНОВКОЮ*".

Ключі: $K_1 = \{5, 3, 1, 2, 4, 6\}$;

$K_2 = \{4, 2, 3, 1\}$.

Матриця складається із шести рядків і чотирьох стовпців 7 × 4 (рис. 1.3):

1) запис у рядках, відповідно до ключа K_1;

2) читання у стовпцях, відповідно до ключа K_2.

1	Н	Я	_	П
2	Е	Р	Е	С
3	У	В	А	Н
4	Т	А	В	Л
5	Ш	И	Ф	Р
6	Е	Н	Н	Я
7	М	_	_	_
K_1/K_2	1	2	3	4

Криптограма: "ПСНЛРЯ_НРВАИН_ЯЕАВФН_НЕУТШЕМ".

Рис 1.3. **Матрична перестановка з подвійним ключем**

Шифри перестановки зберігають усі літери відкритого тексту, але розміщують їх у криптотексті в іншому порядку.

Хоча багато сучасних алгоритмів використовують перестановку, із ними пов'язано проблему необхідності у використанні великого обсягу пам'яті, а також іноді потрібна робота з повідомленнями певного розміру. Тому частіше використовують шифри підстановки.

Скитала (шифр Стародавньої Спарти). Скитала – це один із найдавніших відомих криптографічних пристроїв. Цей прилад використовували для шифрування перестановкою. Він складається з палиці та вузької смужки пергаменту, який обмотується навколо неї спіраллю. Текст, який потрібно зашифрувати, писався на пергаментній стрічці по довжині палиці. Із досягненням краю стрічки палиця поверталася, текст продовжували писати далі доти, поки не закінчувався текст або стрічка. В останньому випадку використовували черговий шматок пергаментної стрічки. Після закінчення шифрування, стрічку знімали з палиці та передавали тим чи тим способом адресатові. Розшифрування виконували з використанням палиці такого самого діаметру. Таким чином, довжину блока n визначали довжиною та діаметром палички, а саме шифрування полягало в перестановці символів вихідного тексту.

Перевага шифру скитала полягає у простоті та відсутності помилок – дуже важлива якість на полі бою. Однак такий шифр можна легко зламати.

Шифри підстановки. *Підстановкою* називають взаємно однозначне відображення *f* деякої кінцевої множини *M* на множину *M′* або на себе. У цьому процесі беруть участь усі елементи множин *M* і *M′*.

Одноалфавітні шифри: шифр простої заміни. У процесі шифрування здійснюють перетворення відкритого тексту *M* довжини *l* таким чином, щоб кожний символ замінювався на деякий інший. Однаковим символам у відкритому тексті відповідають однакові символи криптотексту, а різним – різні. Ключем є таблиця, де встановлено правило заміни символів, тобто кожному символу *a* алфавіту *A* ($a \in A$) ставиться у відповідність символ $b \neq a$ із цього самого алфавіту *A*.

Позначте операцію переходу стрілкою →, тоді вона буде мати такий вигляд:

$$a_i, a_j \in A, a_i \rightarrow a_j, \text{ за } i \neq j, \qquad (1.1)$$

де $i, j \in [0, \ldots, |A|]$; $|A|$ потужність алфавіту, тобто кількість елементів в алфавіті.

Візьміть алфавіт української мови і позначте його як алфавіт A_y (рис. 1.4):

№	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A_y	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м
K	й	ц	у	к	е	н	г	ш	з	щ	х	ґ	ф	ї	в	а	п

№	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
A_y	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	_
K	р	л	о	д	ж	є	я	␣	с	м	и	т	ь	б	ю	ч	і

Рис. 1.4. **Процес шифрування простою заміною**

Відкритий текст: "захист_інформації".

Криптограма: "щйсхжєіґр_лдпймґф".

Шифр Цезаря. Використовуйте алфавіт A_y, розглянутий у шифрі простої заміни (рис. 1.5):

№	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A_y	А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М

№	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
A_y	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	_

Рис. 1.5. **Вигляд алфавіту**

Виберіть секретний ключ *K*, поданий числом із діапазону $1 \le K \le \le |A_{укр.}| - 1$. Як видно, кількість ключів залежить тільки від потужності алфавіту.

Шифрування відкритого повідомлення *M* шифром Цезаря виконують за літерами. Кожну літеру замінюють на таку, розміщеної на *K* символів правіше за літеру, що шифрують. У разі, якщо під час шифрування деякої літери на ключі *K* було досягнуто закінчення алфавіту, то переходьте у його початок і продовжуйте зсув на кількість, що залишилася, символів із ключа.

Приклад:

Шифрують букву "Х" на ключі $K = 20$, тоді криптограма дорівнює "І".

Це перетворення легко описати операцією додавання за модулем потужності алфавіту:

$$c_i = ((J(m_i) + K) \bmod n, \ i = \overline{0, t-1}, \qquad (1.2)$$

де m_i – і-літера відкритого тексту;

$J(m_i)$ – функція, що обчислює порядковий номер літери m_i в алфавіті A;

n – кількість літер в алфавіті;

t – довжина повідомлення в символах.

Розшифровування виконують у зворотному порядку, тобто літера відкритого тексту буде розташовуватися лівіше на *K* символів в алфавіті *A*.

Математично це перетворення подають такою формулою:

$$m_i = (J(c_i) - K) \bmod n, i = \overline{0,t-1}, \qquad (1.3)$$

де c_i – і-літера криптограми;

$J(c_i)$ – функція, що обчислює порядковий номер літери c_i в алфавіті *A*;

n – кількість літер в алфавіті;

t – довжина повідомлення.

Афінна криптосистема. Узагальненням системи Цезаря є афінна криптосистема. Її визначають двома числами a і b, де $0 \le a, b \le n - 1$. Тут n – потужність алфавіту A. Числа a і n мають бути взаємно прості, $НСД\ (a, n) = 1$. Відповідними замінами є:

$$A_{a,b}(j) = (a \times j + b)\ mod\ n; \tag{1.4}$$

$$A^{-1}_{a,b} = (j - b) \times a^{-1}\ mod\ n. \tag{1.5}$$

Зворотну заміну також можна визначити, просто помінявши місцями рядки в таблиці замін.

Взаємна простота a і n необхідна для бієктивності відображення, в іншому випадку можливі відображення різних символів в один, що призведе до неоднозначності дешифрування.

Приклад:

Обчислити обернений елемент до числа $a = 3$ за модулем $n = 11$. Обернений елемент дорівнює $a^{-1} = 4$.

Шифр Цезаря із ключовим словом. У цьому разі ключ K задають числом $k\ (0 \le k \le n - 1)$ і коротким ключовим словом або реченням. Виписують алфавіт, а під ним, починаючи з k-позиції, розміщують ключове слово. Решту літер записують за алфавітом після ключового слова. У результаті визначено потрібну підстановку. Обов'язкова вимога – щоб усі літери ключового слова були різними, не обов'язкова – повторювані літери пропускають. Кількість ключів у системі Цезаря із ключовим словом дорівнює $n!$.

Квадрат Цезаря. Для шифрування квадратом Цезаря необхідно вибирати повідомлення довжиною, кратною квадрату простого числа: 4, 9, 16, 25, 36, 49 тощо. В оригіналі використовували повідомлення довжиною 25 символів.

Приклад:

Потрібно зашифрувати:

Love is blind, as well as hatred.

Приберіть усі пробіли та розділові знаки:

l o v e i s b l i n d a s w e l l a s h a t r e d

Тепер усі літери занесіть до квадратної таблиці зі стороною, що дорівнює квадрату довжини повідомлення. У цьому разі довжина рядка дорівнює 25, тому таблиця має розмір 5 × 5. Літери розташовуйте у стовпчику, одна за одною (рис. 1.6).

l	s	d	l	a
o	b	a	l	t
v	l	s	a	r
e	i	w	s	e
i	n	e	h	d

Рис. 1.6. **Квадратна таблиця з літерами**

На наступному етапі перепишіть літери з таблиці по рядках:

l s d l a o b a l t v l s a r e i w s e i n e h d

Розшифрування відбувається в тому самому алгоритмі.

Шифр Трітеміуса. Шифр Трітеміуса вважають удосконаленим шифром Цезаря, тобто це шифр підстановки. За алгоритмом шифрування кожен символ повідомлення замінюють на символ, розміщений лівіше за цей на певну величину (крок зміщення). Тут крок зміщення стає змінним, тобто залежить від будь-яких додаткових факторів. Наприклад, можна задати закон зміщення (рівняння шифрування) у вигляді лінійної функції від позиції літери, яку шифрують. Сама функція має гарантувати цілочисельне значення. Пряма функція шифрування повинна мати обернену, що також забезпечує цілочисельне значення.

Рівняння шифрування для шифру Трітеміуса має такий вигляд:

$$L = (m + k) \bmod N, \qquad (1.6)$$

де L – номер зашифрованої літери в алфавіті;

m – номер позиції чергової літери відкритого тексту в алфавіті;

k – крок зміщення (функціональна залежність від позиції літери в повідомленні);

N – потужність алфавіту.

Деякі варіанти обчислення кроку зміщення k:

$$k = A_i + B; \tag{1.7}$$

$$k = A_i^2 + B_i + C, \tag{1.8}$$

де i – позиція літери в повідомленні;

A, B, C – ключі.

Шифр Плейфера. Цей шифр працює з парами літер, біграмами, які замінюють за спеціальними правилами.

Увесь алфавіт розташовано в матриці. Відкритий текст M розподіляють на пари символів m_i, m_{i+1}. Кожну пару символів відкритого тексту замінюють на пару символів із матриці таким чином:

1) якщо символи розміщено в одному рядку, то кожен із символів пари замінюють на наступний, розташований праворуч від нього (за останнім символом у рядку слідує перший);

2) якщо символи розміщено в одному стовпчику, то кожен символ пари замінюють на символ, розташований нижче від нього (за останнім нижнім символом слідує верхній);

3) якщо символи пари розміщено в різних рядках і стовпцях, то їх уважають протилежними кутами прямокутника. Символ, розташований у лівому куті, замінюють на символ, що стоїть в іншому лівому куті; заміну символу, розташованому у правому куті, здійснюють аналогічно;

4) якщо літери пари однакові, то перед шифруванням між ними вставляють спеціальний символ (наприклад, тире) (рис. 1.7).

Відкритий текст: "ШИФР_ПЛЕЙФЕРА_".

А	Х	Б	М	Ц	В
Ч	Г	Н	Ш	Д	О
Е	Щ	,	Х	У	П
.	З	Ъ	Р	И	Й
С	Ь	К	Э	Т	Л
Ю	Я		Ї	Ф	_

ШИ	РД
ФР	ИЇ
_П	ВЙ
ЛЕ	П.
ЙФ	_И
ЕР	.Х
А_	ЮВ

Рис 1.7. **Демонстрація шифру Плейфера**

Шифротекст: "РДИЇВЙП.–И.ХЮВ"

Дошка (квадрат) Полібія. У криптографії квадрат Полібія, також відомий як шахова дошка Полібія, – одна з найдавніших систем шифрування, запропонована грецьким істориком і державним діячем Полібієм.

Для кожної мови окремо будують таблицю шифрування з однаковою (не обов'язково) кількістю пронумерованих рядків і стовпців. Параметри таблиці залежать від її потужності (кількості літер в алфавіті). Беруть два цілих числа, добуток яких найближчий до кількості літер у мові, визначають потрібну кількість рядків і стовпців. Потім у таблицю вписують усі літери алфавіту одна за одною – по одній на кожну клітинку. У разі нестачі клітинок можна вписати в одну клітинку дві літери (які, наприклад, рідко вживаються або схожі за вживанням).

Для шифрування на квадраті беруть літеру з тексту, а в шифротекст заносять пару чисел – номерів рядка та стовпчика, на перетині яких розміщено цю літеру.

Шифр Вітстона. 1854 року Чарльз Вітстон розробив новий метод шифрування біграм, який називають подвійним квадратом. Свою назву цей шифр дістав за аналогією з полібіанським квадратом. На відміну від полібіанського, шифр "подвійний квадрат" використовує відразу дві таблиці, розміщені на одній горизонталі, а шифрування виконують біграмами (парами), як у шифрі Плейфейра. Шифр "подвійний квадрат" виявився дуже надійним і зручним та застосовувався Німеччиною в роки Другої світової війни.

Перед шифруванням вихідне повідомлення розподіляють на біграми. Кожну біграму шифрують окремо. Першу літеру знаходять у лівій таблиці, а другу – у правій. Подумки будують прямокутник так, щоб літери біграми були розміщені на його протилежних вершинах. Інші дві вершини цього прямокутника дають літери шифротексту.

Якщо обидві літери біграми повідомлення розташовано в одному рядку, то й літери шифротексту беруть із цього ж рядка. Першу беруть із лівої таблиці у стовпці, номер якого відповідає другій літері біграми повідомлення. Другу ж літеру біграми шифротексту беруть із правої таблиці у стовпці, номер якого відповідає першій літері біграми повідомлення.

Шифр Ґронсфельда. Цей шифр використовує числовий ключ, а сама схема нагадує шифр Цезаря. Нехай нам потрібно зашифрувати слово EXALTATION. Наприклад, беріть як ключ число *k = 31 415*, потім будуйте табл. 1.1:

Таблиця 1.1

Таблиця шифрування

E	X	A	L	T	A	T	I	O	N
3	1	4	1	5	3	1	4	1	5

Кожній літері відповідає певна цифра, ця цифра буде показувати, на скільки позицій буде відбуватися зміщення алфавіту для кожної конкретної літери. Наприклад, літері *E* відповідає *H*. Таким чином, для всього слова визначають зашифрований текст: HYEMYDUMPS. Обернене перетворення виконують подібним чином, тільки зміщення відбувається в інший бік.

Частотний аналіз. Принцип частотного криптоаналізу побудовано на різній частоті використання в текстах різних літер алфавіту. Розподіл літер у криптотексті порівнюють з алфавітом вихідного повідомлення. Літеру, яку найчастіше зустрічають у криптотексті, замінюють тією, що найчастіше вживають у цій мові (дані беруть зі спеціально побудованих частотних таблиць). Імовірність успішного розкриття підвищується зі збільшенням довжини криптотексту. Є безліч різних таблиць із частотним розподілом літер у тій чи тій мові, але жодна з них не містить остаточної інформації, навіть порядок літер у них може відрізнятися. Більш того, частотний розподіл літер значно залежить від типу тексту: проза, усне мовлення, технічна мова тощо. Розгляньте табл. 1.2.

Таблиця 1.2

Таблиці розподілу літер у різних мовах

Літера	Частота	Літера	Частота	Літера	Частота
В українській мові					
1	2	3	4	5	6
А	0.070	Л	0.028	Ц	0.009
Б	0.010	М	0.033	Ч	0.011
В	0.046	Н	0.070	Ш	0.005
Г	0.013	О	0.082	Щ	0.003
Д	0.028	П	0.025	Ы	0.016
Е	0.043	Р	0.038	ь	0.015
Ж	0.008	С	0.036	Э	0.006
З	0.019	Т	0.051	Ю	0.009
И	0.056	У	0.027	Я	0.021
Й	0.007	Ф	0.005	–	0.134
К	0.036	Х	0.010	І	0.037
Ї	0.006	Ґ	0.000		

Закінчення табл. 1.2

1	2	3	4	5	6
У російській мові					
А	0.062	Л	0.035	Ц	0.004
Б	0.014	М	0.026	Ч	0.012
В	0.038	Н	0.053	Ш	0.006
Г	0.013	О	0.090	Щ	0.003
Д	0.025	П	0.023	Ы	0.016
Е	0.072	Р	0.040	Ъ, Ь	0.014
Ж	0.007	С	0.045	Э	0.003
З	0.016	Т	0.053	Ю	0.006
И	0.062	У	0.021	Я	0.018
Й	0.010	Ф	0.002	–	0.174
К	0.028	Х	0.009		
В англійській мові					
A	0.0804	B	0.0154	C	0.0306
D	0.0399	E	0.1251	F	0.0230
G	0.0196	H	0.0549	I	0.0726
J	0.0016	K	0.0067	L	0.0414
M	0.0253	N	0.0709	O	0.0760
P	0.0200	Q	0.0011	R	0.0612
S	0.0654	T	0.0925	U	0.0271
V	0.0099	W	0.0192	X	0.0019
Y	0.0173	Z	0.0009		

Хоча немає таблиці, яка може врахувати всі види текстів, але є загальне для всіх таблиць, наприклад, в англійській мові літера *E* завжди очолює список частот; *T* має другу позицію; *A* і *O* майже завжди – треті. Крім того, дев'ять літер англійської мови *E*, *T*, *A*, *O*, *N*, *I*, *S*, *R*, *H* завжди мають частоту, вищу за інші. Ці дев'ять літер входять до складу приблизно 70 % слів англійських текстів. Найпоширеніші літери європейських мов наведено в табл. 1.3.

Таблиця 1.3

Таблиця найвагоміших (найбільш поширених) літер у різних мовах

Мови											
російська		англійська		німецька		французька		італійська		фінська	
Літера (Л)	**Частота (Ч)**	**Л**	**Ч**	**Л**	**Ч**	**Л**	**Ч**	**Л**	**Ч**	**Л**	**Ч**
о	0.1090	e	0.1251	e	0.1846	e	0.1587	e	0.1179	a	0.1206
е	0.0872	t	0.0925	n	0.1142	a	0.0942	a	0.1174	i	0.1059
а	0.0751	a	0.0804	i	0.0802	i	0.0841	i	0.1128	t	0.0976
и	0.0751	o	0.0760	r	0.0714	s	0.0790	o	0.0983	n	0.0864
н	0.0642	i	0.0726	s	0.0704	t	0.0726	n	0.0688	e	0.0811
т	0.0642	n	0.0709	a	0.0538	n	0.0715	l	0.0651	s	0.0783
с	0.0545	s	0.0654	t	0.0522	r	0.0646	r	0.0637	l	0.0586
р	0.0484	r	0.0612	u	0.0501	u	0.0624	t	0.0562	o	0.0554
в	0.0460	h	0.0549	d	0.0494	l	0.0534	s	0.0498	k	0.0520
∑	**0.6235**	∑	**0.6990**	∑	**0.7263**	∑	**0.7405**	∑	**0.7500**	∑	**0.7359**

Зауважте, що літери *I*, *N*, *S*, *E*, *A* (*I*, *H*, *C*, *E*, *A*) належать до високочастотного класу кожної мови. Також є таблиці частоти появи літер на початку, у кінці слова, уживання пар літер тощо.

Найпростіший захист від частотних атак забезпечено системою омофонів (homophones) – однозвучних підстановлювальних шифрів, у яких один символ відкритого тексту відображається як кілька символів шифротексту; їхня кількість пропорційна частоті появи літери. Шифруючи літеру вихідного повідомлення, виберіть випадково одну з її замін. Отже, простий підрахунок частот нічого не дає криптоаналітику. Однак доступна інформація про розподіл пар і трійок літер у різних природних мовах. Криптоаналіз із використанням такої інформації буде більш успішним.

Богатоалфавітні шифри (поліалфавітні) було винайдено Ліном Баттістою (Lean Battista) 1568 року. Основна ідея багатоалфавітних систем полягає в тому, що протягом усього тексту ту саму літеру може бути зашифровано по-різному. Таким чином, заміни для неї вибирають із багатьох алфавітів, залежно від положення в тексті. Це є гарним захистом від простого підрахунку частот, тому що немає єдиного маскування для кожної літери у криптотексті. У таких шифрах використовують множину ключів, кожен із яких застосовують для шифрування одного символу

відкритого тексту. Першим ключем шифрують перший символ відкритого тексту, другим – другий і т. д. Після використання всіх ключів їх повторюють циклічно.

Шифр Віженера. Однією з найбільш відомих багатоалфавітних криптосистем є система Віженера, названа на честь французького дипломата та криптографа Блеза де Віженера (Vigenere). Цей метод було уперше опубліковано 1586 року. У цьому шифрі ключ K задано набором з d літер, що належать алфавіту A:

$$K = k_0 k_1 \ \ldots \ k_{d-1}, \tag{1.9}$$

де $k_i \in A$.

Набори підписують із повторенням під відкритим повідомленням M:

$$M = m_0 m_1 \ \ldots \ m_{d-1} m_d \ \ldots \ m_t; \tag{1.10}$$

$$K = k_0 k_1 \ \ldots \ k_{d-1} k_d \ \ldots \ k_{t\,(mod\ d)}. \tag{1.11}$$

Знайдену послідовність додають до відкритого тексту за модулем потужності алфавіту $n = |A|$:

$$c_i(M^t, K^d) = (J(m_i) + J(k_{i\ mod\ d}))\ (mod\ n),\ i = \overline{1,t}. \tag{1.12}$$

Замість формули часто використовують таблицю, де літеру шифротексту знаходять на перетині стовпчика, у заголовку якого стоїть літера відкритого тексту, та рядка з літерою ключа в заголовку (табл. 1.4). Якщо ключ K складається з одного символу $d = 1$, то буде визначено класичний шифр Цезаря.

Шифр Віженера з автоключем за відкритим текстом. Подальшою модифікацією системи Віженера є система шифрів із *автоключем (Autokey),* що приписують математикові XVI с. Дж. Кардано, Autoclave. Шифрування починають за допомогою "первинного ключа" (який є справжнім ключем) та доповнюють із використанням повідомлення, зміщеного на довжину первинного ключа; потім виконують додавання за модулем n, який дорівнює потужності алфавіту A.

Таблиця 1.4

Приклад таблиці Віженера (для російського алфавіту)

	Буквы открытого текста																														
Буквы ключа	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Перетворення відкритого тексту у криптограму здійснюють за такою формулою:

$$c_i(M^l, K^d) = \begin{cases} \left(J(m_i) + J(k_i)\right) \bmod n, & i < d: \\ \left(J(m_i) + J\left(m_{i-d}\right)\right) \bmod n, & i \geq d. \end{cases} \quad (1.13)$$

Розшифрування виконують за такою формулою:

$$m_i(C^l, K^d) = \begin{cases} \left(J(c_i) + J(k_i)\right) \bmod n, & i < d: \\ \left(J(c_i) + J\left(m_{i-d}\right)\right) \bmod n, & i \geq d. \end{cases} \quad (1.14)$$

Розшифрування не складне: за первинним ключем K^d розшифровують d перших символів повідомлення, після чого знайдену частину вихідного повідомлення використовують як ключ. Другу частину процесу розшифрування повторюють доти, доки не буде досягнуто кінець криптограми.

Шифр Віженера з автоключем за криптограмою. Як і в попередньому алгоритмі, шифрування здійснюють із застосуванням "первинного ключа" K, а продовжують за допомогою визначеної криптограми c_0, c_1, … з використанням "первинного ключа". Перетворення відкритого тексту на криптограму здійснюють за такою формулою:

$$C_i^*\left(M^l, K_j^{d_j}\right) = \begin{cases} m_i + k_i \bmod n, & i < d: \\ m_i + k_{i-d} \bmod n, & i \geq d. \end{cases} \quad (1.15)$$

Розшифрування виконують так:

$$M_i^*\left(C^l, K_j^{d_j}\right) = \begin{cases} c_i + k_i \bmod n, & i < d: \\ c_i + k_{i-d} \bmod n, & i \geq d. \end{cases} \quad (1.16)$$

Поліалфавітна заміна. Цей клас шифрів об'єднав два види математичних перетворень: перестановку f і підстановку g. За допомогою операції перестановки формують першу частину ключа шифру, ключова матриця підстановки, що складається з кількох різних функцій f_i. Кожна функція є перестановкою алфавіту A. Формування ключової матриці підстановки здійснюють випадковим способом.

Чотири функції f_0, f_1, f_2, f_3 наведено в табл. 1.5.

Таблиця 1.5

Таблиця з функціями

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A_y	А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М
$f_0(x)$	й	ц	у	к	е	н	г	ш	щ	з	х	ї	ф	і	в	а	п
$f_1(x)$	м	и	т	ь	б	ю	й	ц	у	к	е	н	ґ	ш	щ	з	х
$f_2(x)$	р	о	л	д	ж	є	я	ч	с	м	и	т	ь	б	ю	й	ц
$f_3(x)$	ш	щ	ґ	х	_	ф	і	в	а	п	р	о	л	д	м	и	т
A_y	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
$f_0(x)$	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	_
$f_1(x)$	р	о	л	д	ж	є	я	ч	с	м	и	т	ь	б	ю	_	ґ
$f_2(x)$	_	ф	і	в	а	п	р	о	л	д	ж	є	я	ч	с	ї	г
$f_3(x)$	у	к	е	н	г	ш	щ	з	х	ґ	ф	_	в	а	п	і	ї

Друга частина ключа – це функція підстановки g, що визначає, за допомогою якої функції f будуть шифрувати символи відкритого тексту. Довжина функції g може бути довільною. Наприклад: для чотирьох функцій f_i, $i = \overline{0,l}$, функція підстановки може дорівнювати $g = \{1, 3, 0\}$ або $g = \{1, 0, 3, 2, 0, 3\}$. Найоптимальніше її вибрати кратною довжині повідомлення.

Таким чином, ключ шифрування K подають перестановками f_0, f_1, f_2, f_3 і підстановкою $g = \{1, 3, 0, 2, 3\}$ довжиною $s = 5$.

Шифрування повідомлення виконують за таким правилом:

$$c_i\left(M^t, F, g^s\right) = f_{g(i \bmod s)}(m_i), \tag{1.17}$$

а розшифровування – так:

$$m_i\left(C^t, F, g^s\right) = f_{g(i \bmod s)}(c_i). \tag{1.18}$$

Шифрувальна машина "Енігма". "*Енігма*" – це портативна шифрувальна машина, що використовували для шифрування та дешифрування

секретних повідомлень. Точніше, "Енігма" – це ціла сім'я електромеханічних роторних машин, які застосовували із 20-х рр. XX ст.

"Енігму" використовували в комерційних цілях, а також у військових і державних службах багатьох країн світу, але найбільшого поширення набула в нацистській Німеччині під час Другої світової війни. Саме "*Енігма*" вермахту (*WehrmachtEnigma*) – німецька військова модель – найчастіше є предметом дискусій.

Сутність шифру полягає в циклічному застосуванні декількох моноалфавітних шифрів до певної кількості літер, що шифрують. Наприклад, нехай є деяке повідомлення $x_1, x_2, x_3, \ldots x_n, \ldots x_{2n}, \ldots$, яке треба зашифрувати. Під час використання поліалфавітних шифрів є кілька моноалфавітних шифрів (наприклад, n). У цьому разі до першої літери застосовують перший моноалфавітний шифр, до другої літери – другий, до третьої – третій ..., до n-ї літери – n-й, а до $(n + 1)$ – знову перший і т. д. Таким чином, створюється складна послідовність, яку не так просто розкрити, як один моноалфавітний шифр. Найважливішим ефектом, що досягають використанням поліалфавітних шифрів, є маскування частот появи тих чи тих літер у тексті, за допомогою яких зазвичай дуже легко розкривають моноалфавітні шифри.

1.4. Практичне виконання

Перестановний шифр із ключовим словом:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.8).

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.9).

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис 1.8).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

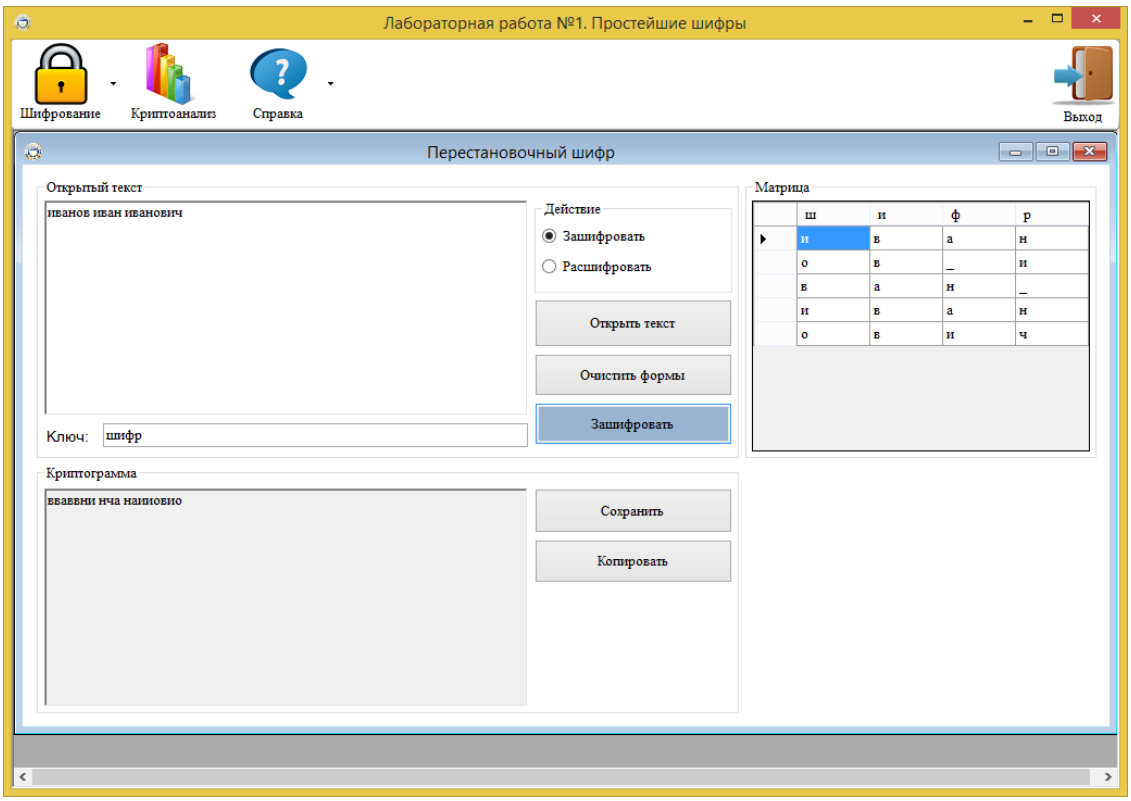


Рис. 1.8. **Зашифрований текст за допомогою перестановного шифру**

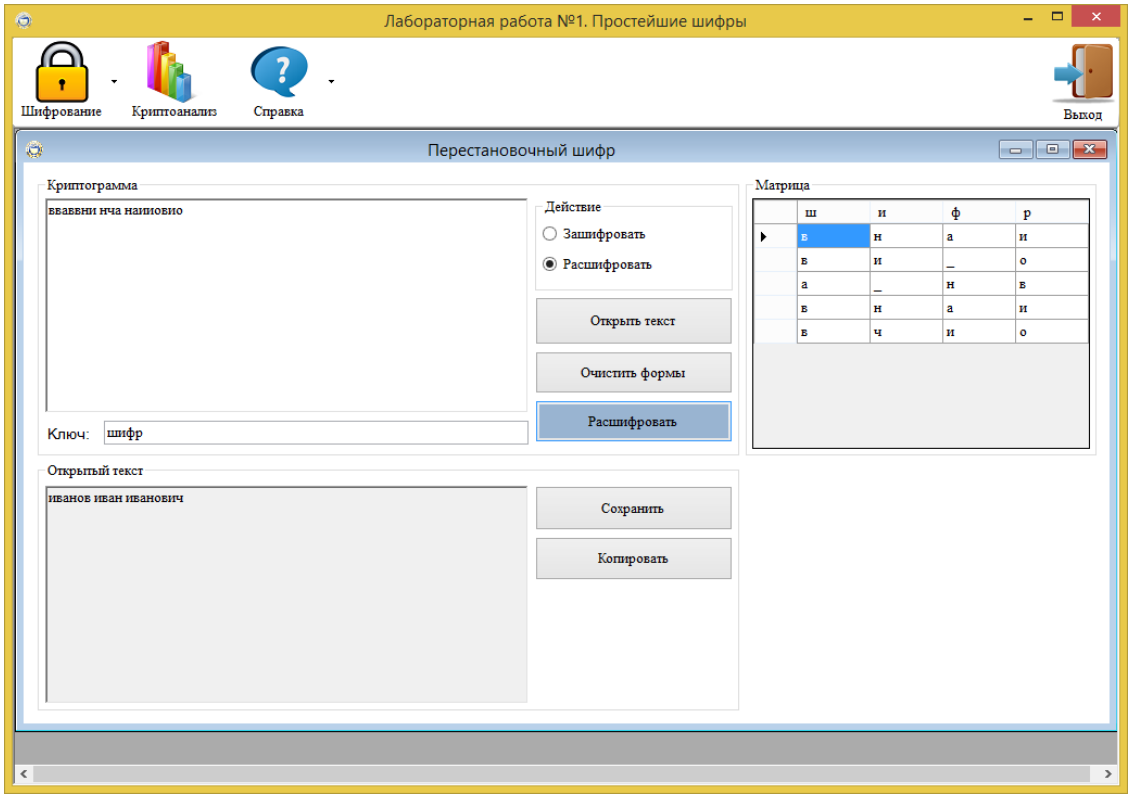


Рис. 1.9. **Розшифрування шифротексту**

Перестановний шифр із подвійним ключем:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрити його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть два ключі (вони мають складатися з малих літер російського алфавіту, також можуть мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.10).

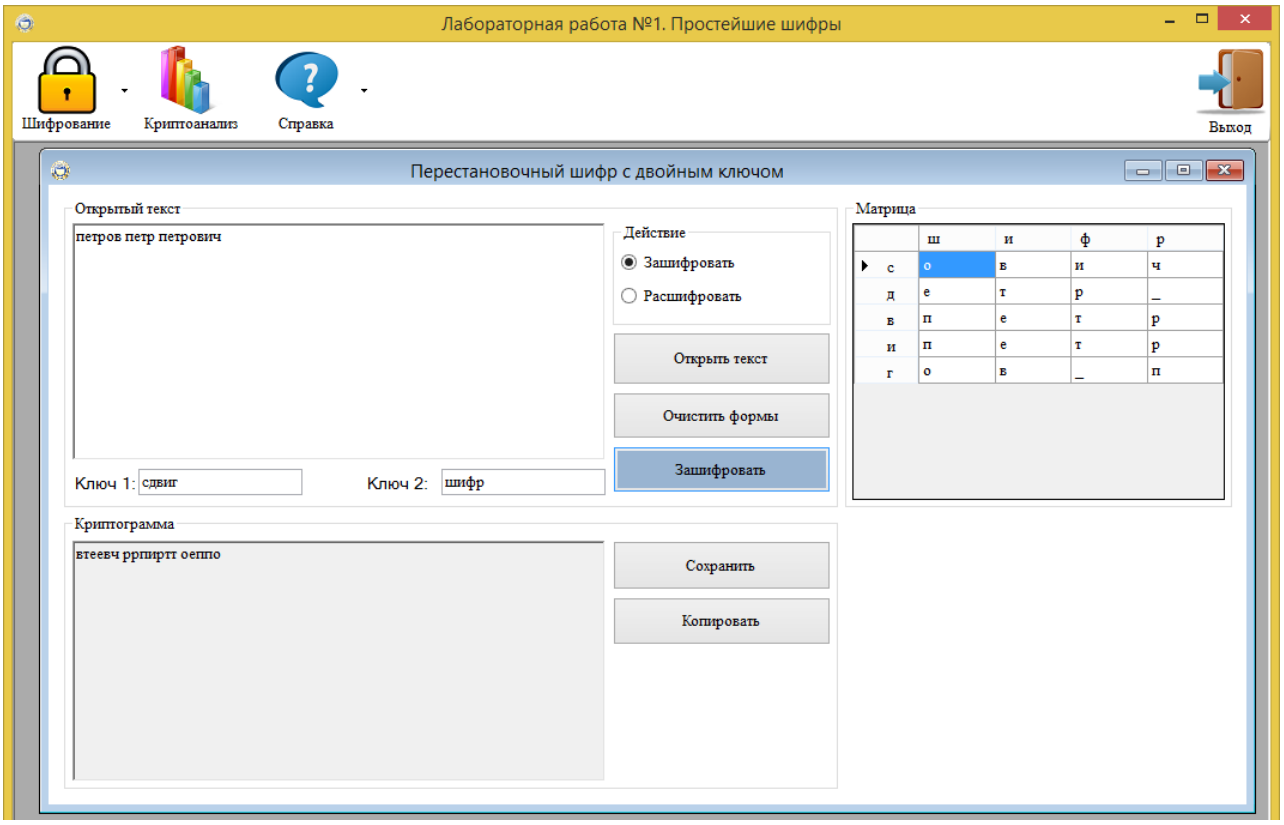


Рис. 1.10. **Зашифрований текст за допомогою перестановного шифру з подвійним ключем**

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.11).

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.10).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

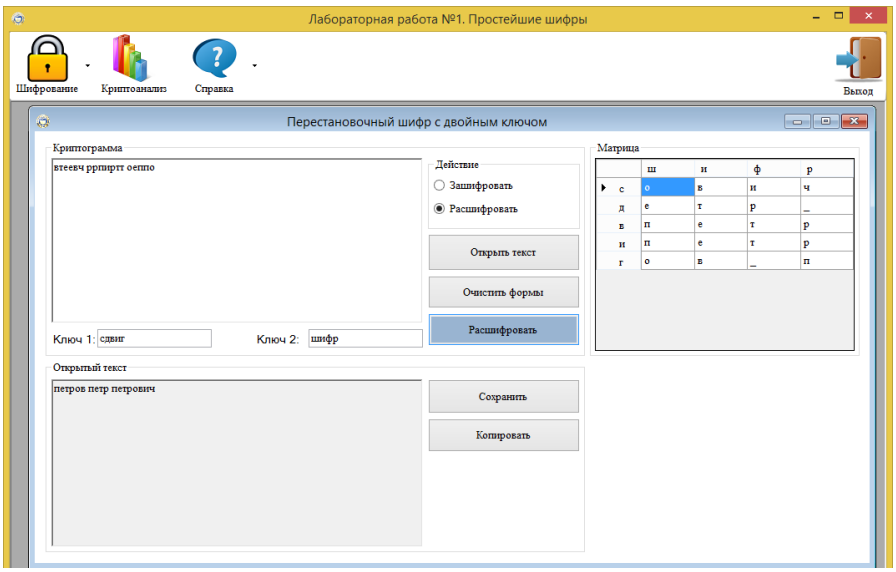


Рис. 1.11. **Розшифрування шифротексту**

Скитала (шифр Стародавньої Спарти):

1. Уведіть ПІБ як відкритий текст (текст має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.12).

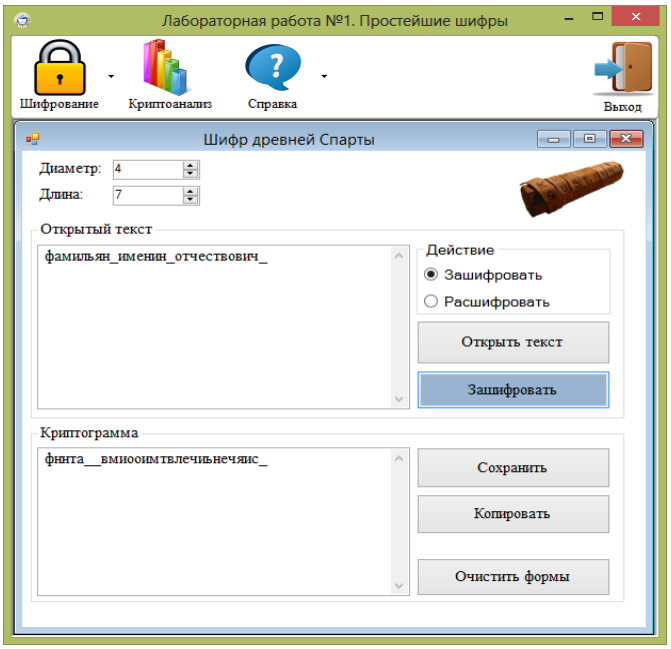


Рис. 1.12. **Зашифрований текст за допомогою шифру Стародавньої Спарти**

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.13).

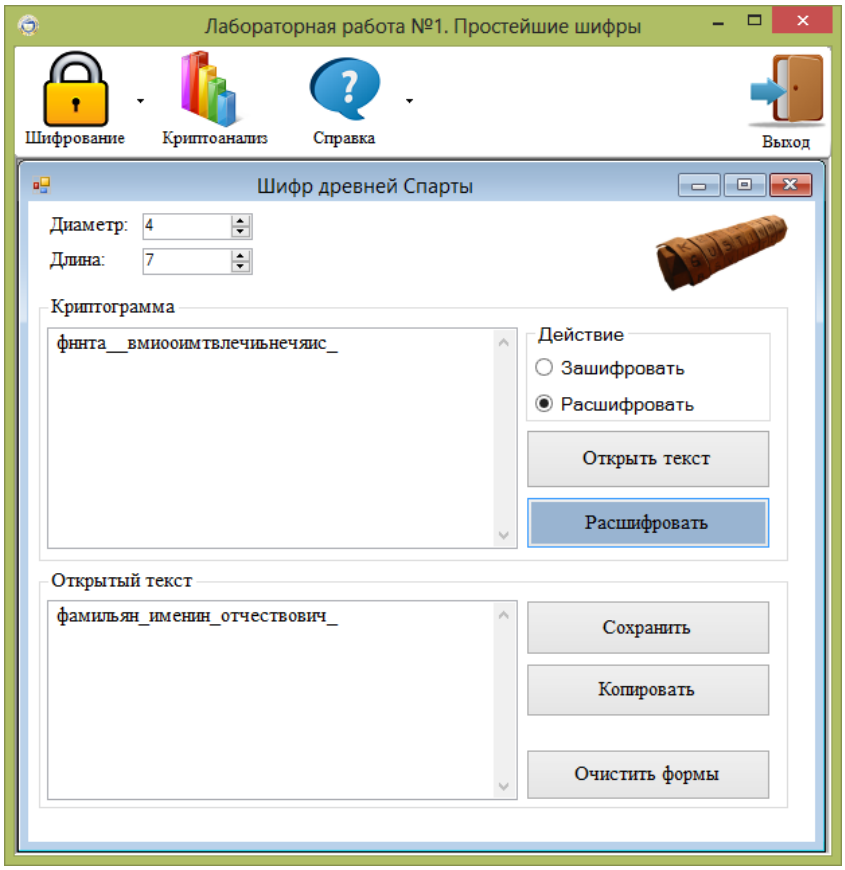


Рис. 1.13. **Розшифрування шифротексту**

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.12).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр підстановки:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли):

- для заповнення ключа автоматично можна натиснути кнопку "Случайный ключ" або заповнити вручну;
- для видалення ключа – "Удалить ключ" (рис. 1.14).

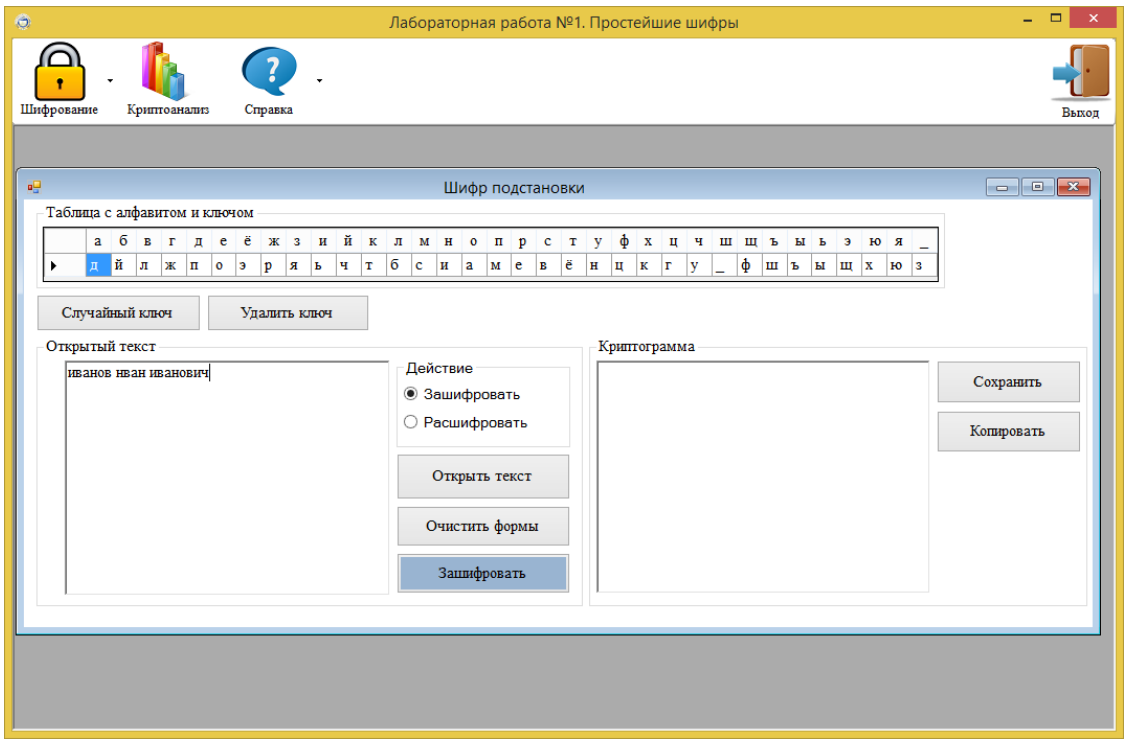


Рис. 1.14. **Додавання ключа**

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.15).

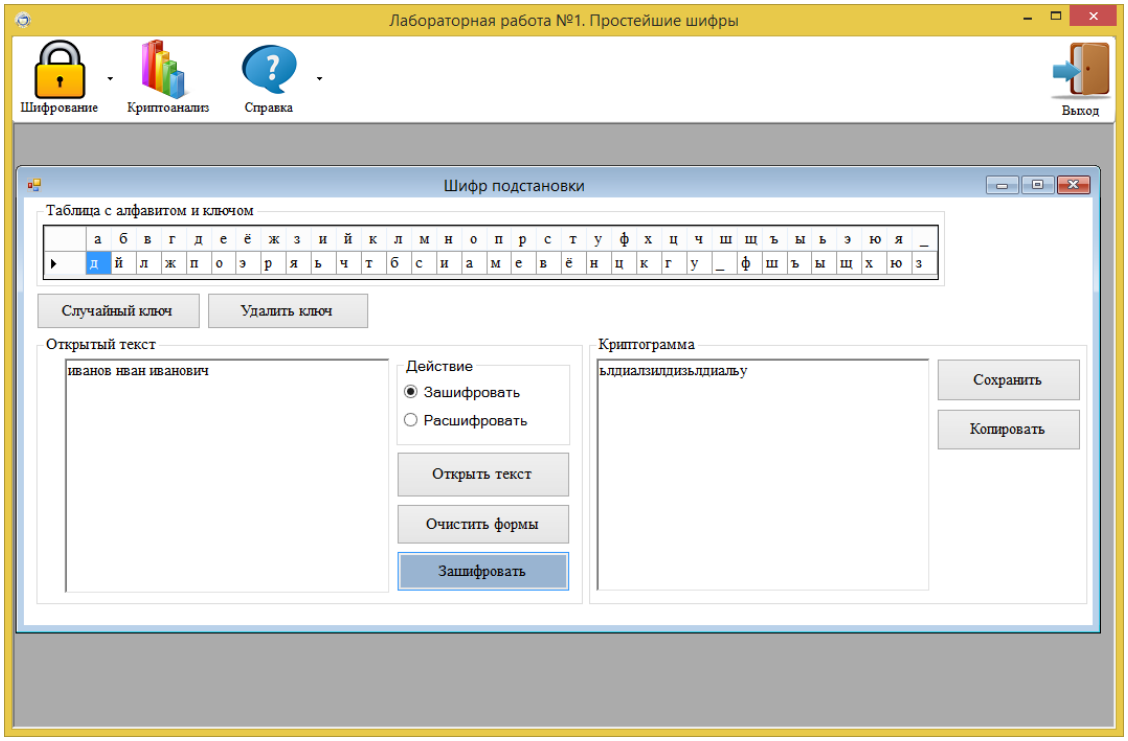


Рис. 1.15. **Зашифрований текст за допомогою шифру підстановки**

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.16).

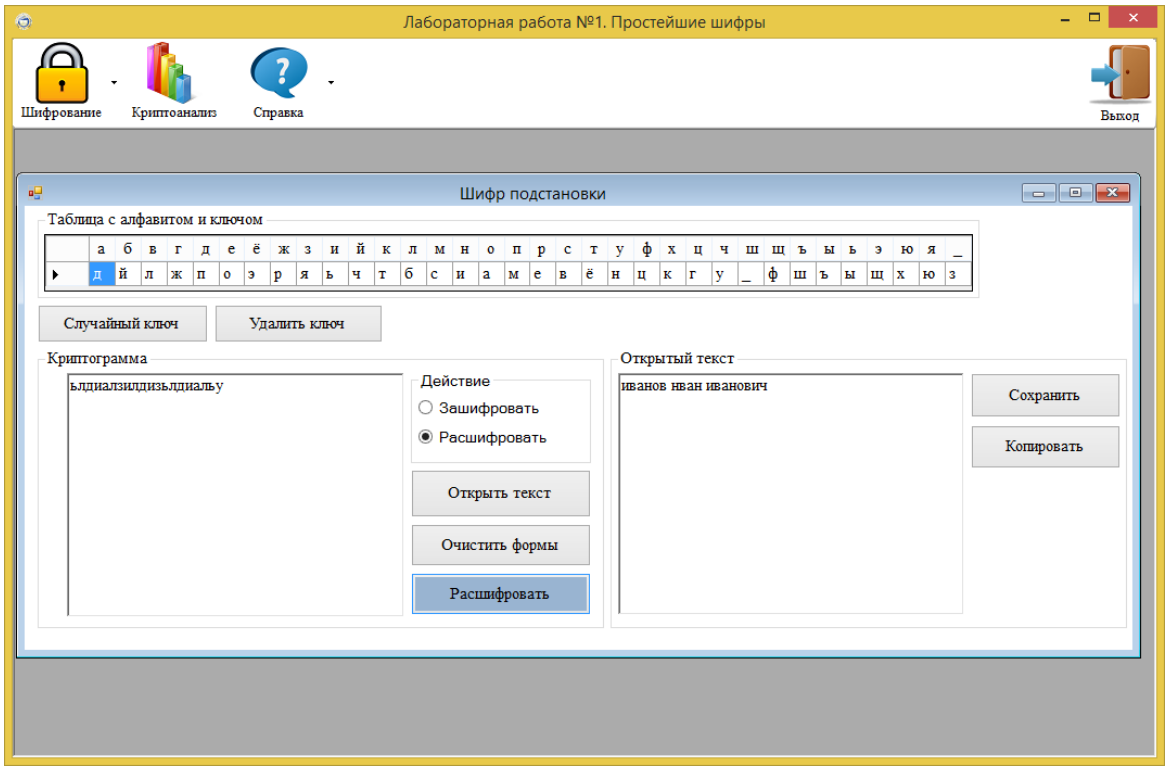


Рис. 1.16. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.15).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Шифр Цезаря:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Увести *k* – зміщення символів.

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.17).

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.18).

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.17).

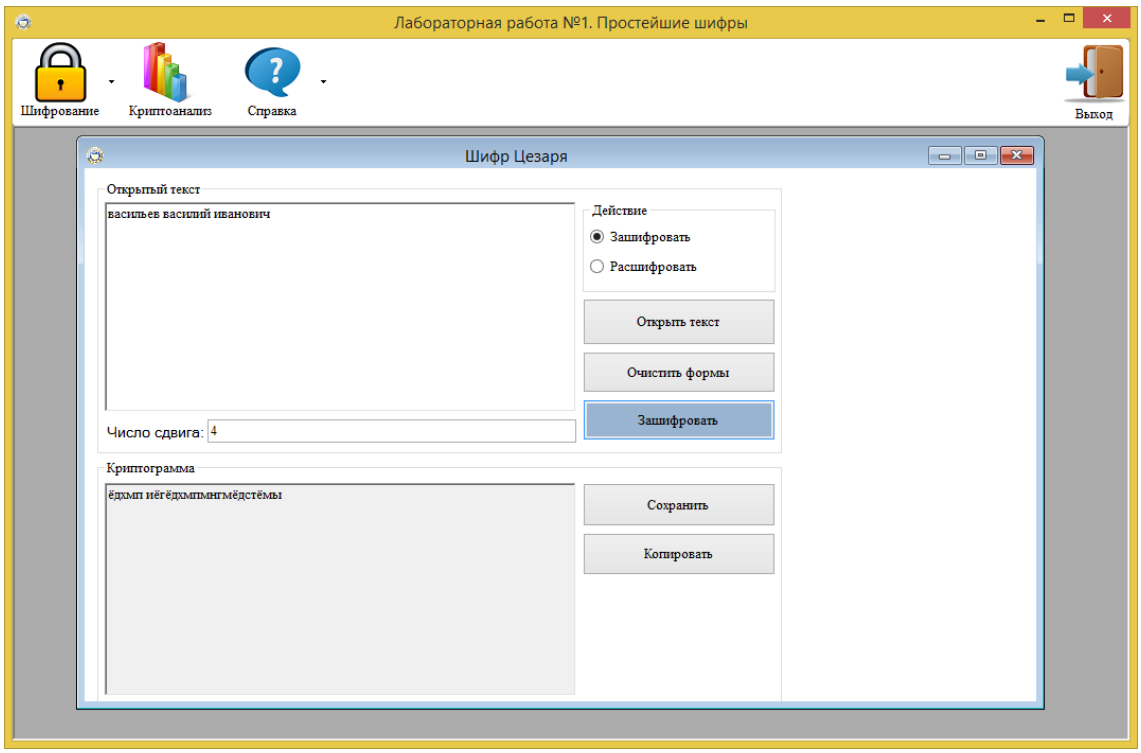


Рис. 1.17. **Зашифрований текст за допомогою шифру Цезаря**

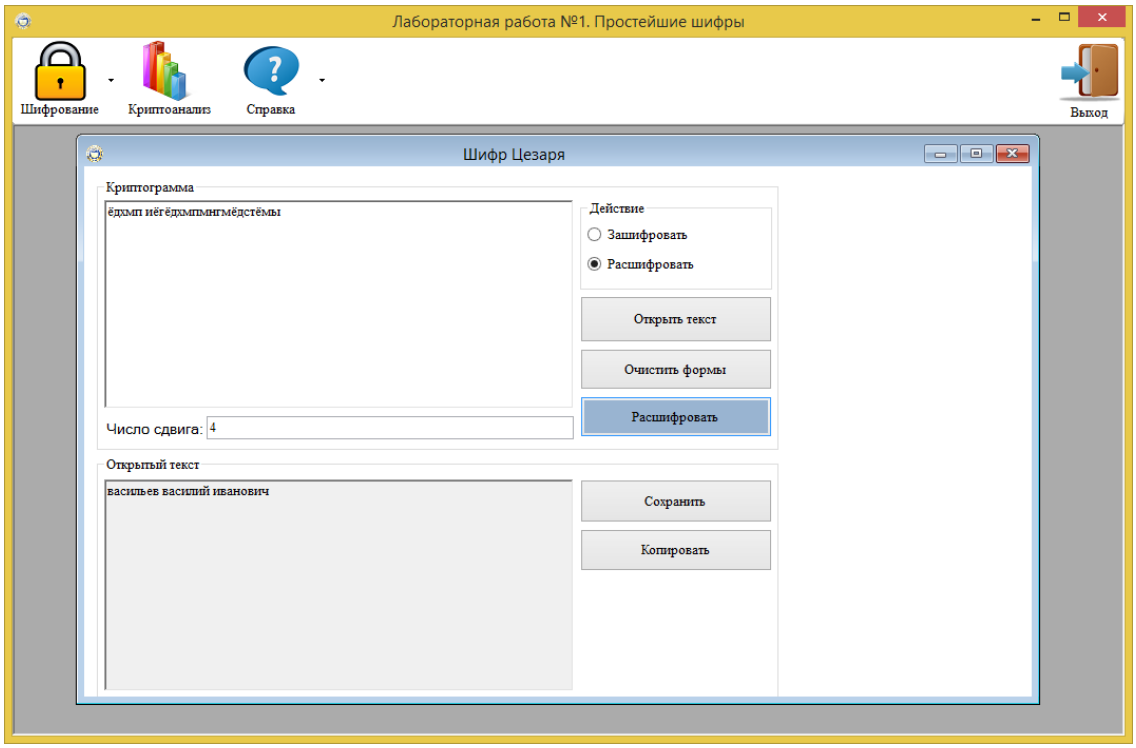


Рис. 1.18. **Розшифрування шифротексту**

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Афінна криптосистема:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту (рис. 1.19), також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

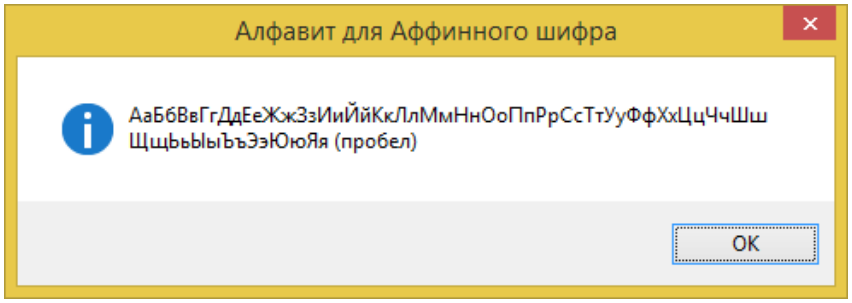


Рис. 1.19. **Алфавіт для Афінної криптосистеми**

2. Уведіть число *A* і зміщення *K*.

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.20).

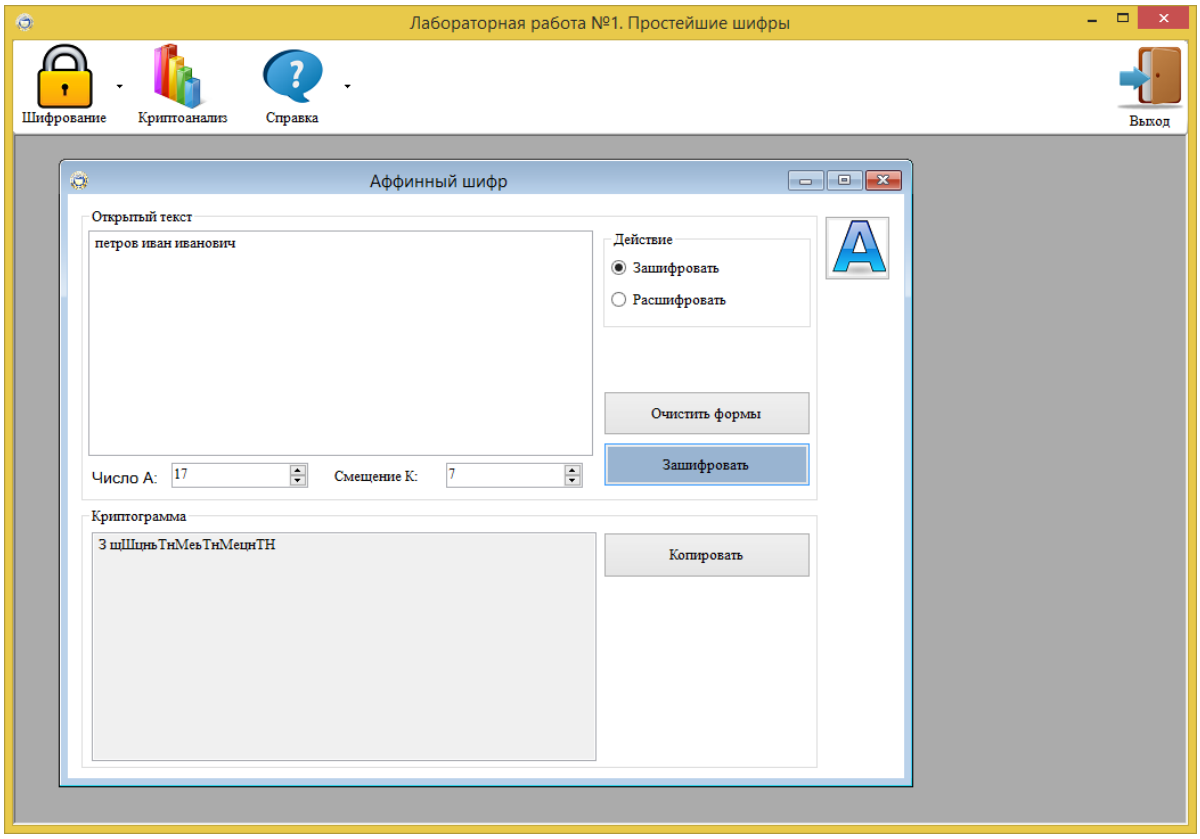


Рис. 1.20. **Зашифрований текст за допомогою афінної криптосистеми**

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.21).

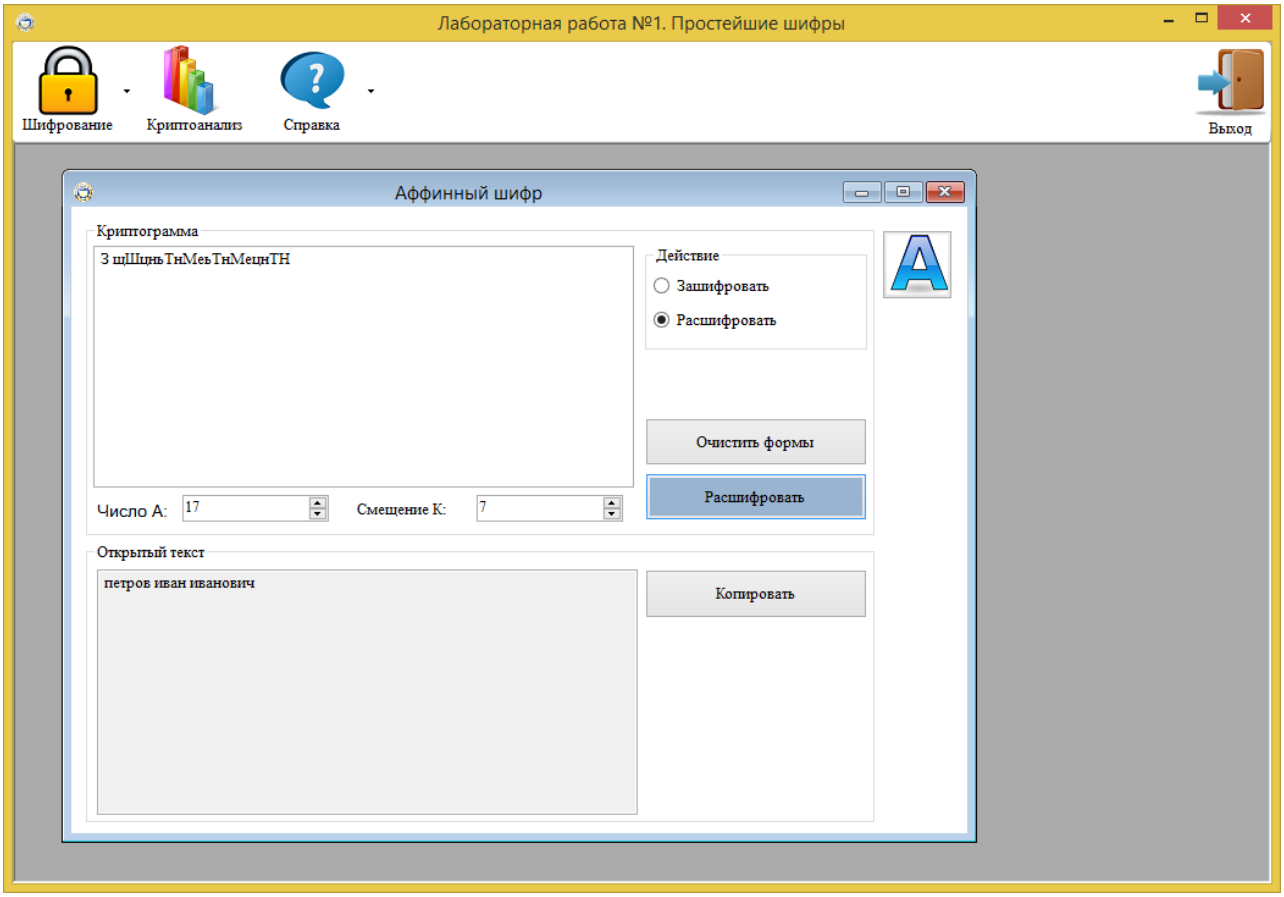


Рис. 1.21. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.20).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Шифр Цезаря із ключовим словом:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.22).

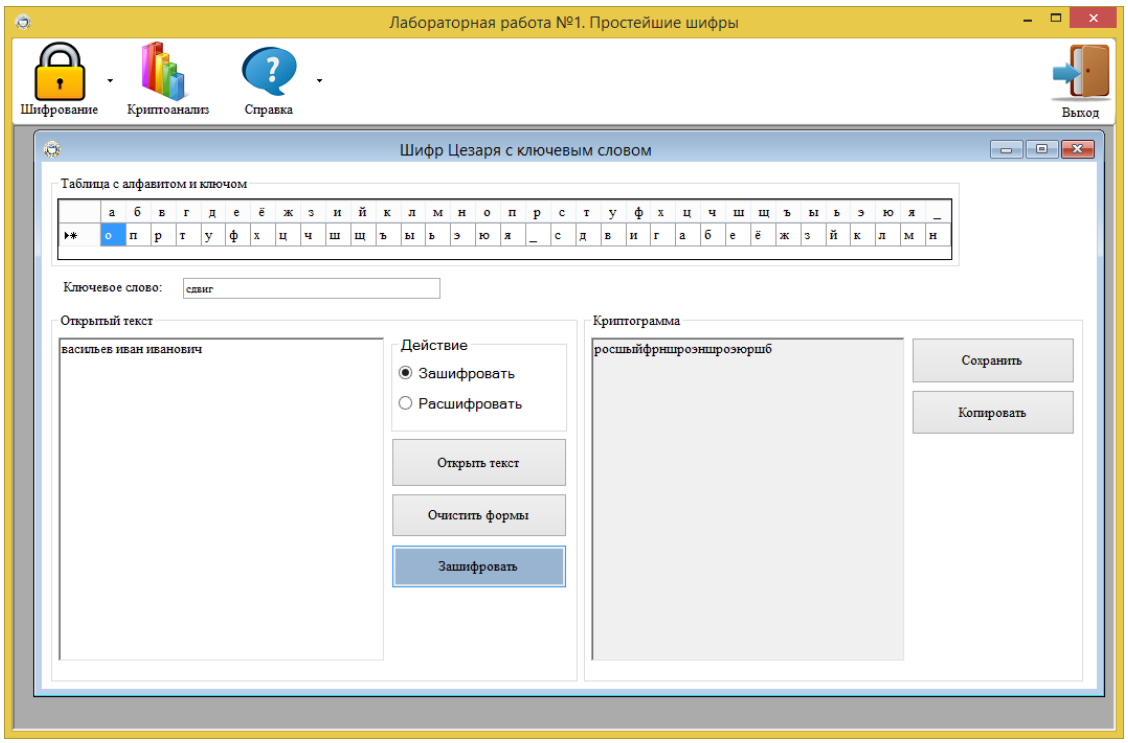


Рис. 1.22. **Зашифрований текст за допомогою шифру Цезаря із ключовим словом**

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.23).

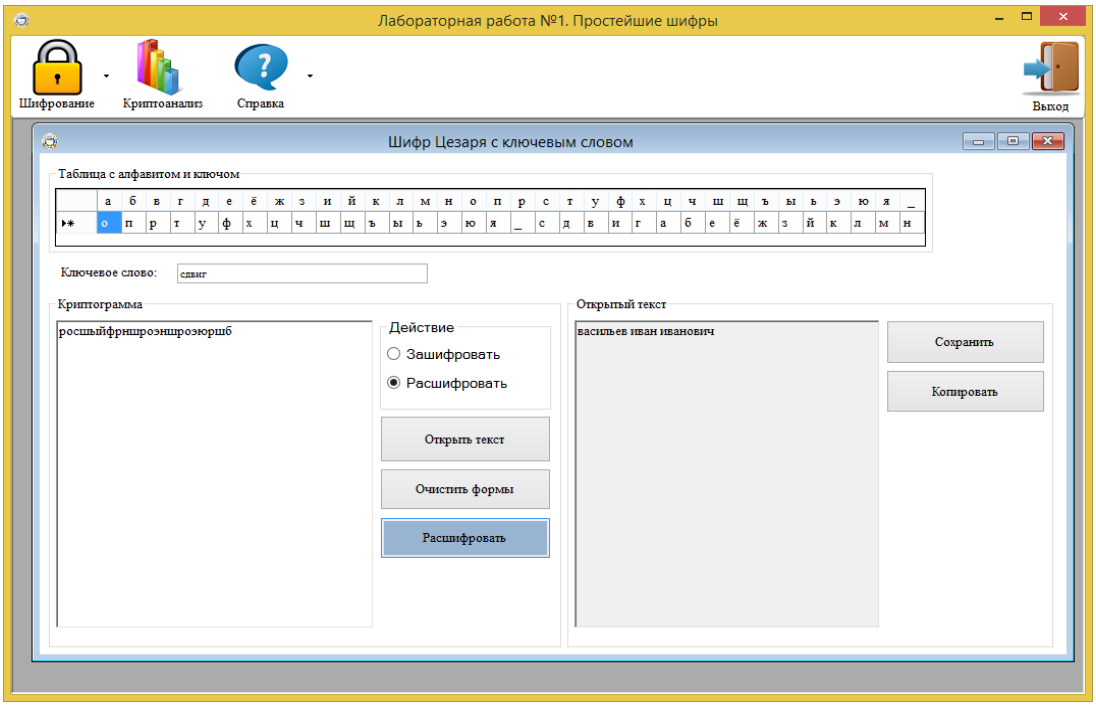


Рис. 1.23. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.22).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Квадрат Цезаря:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.24).

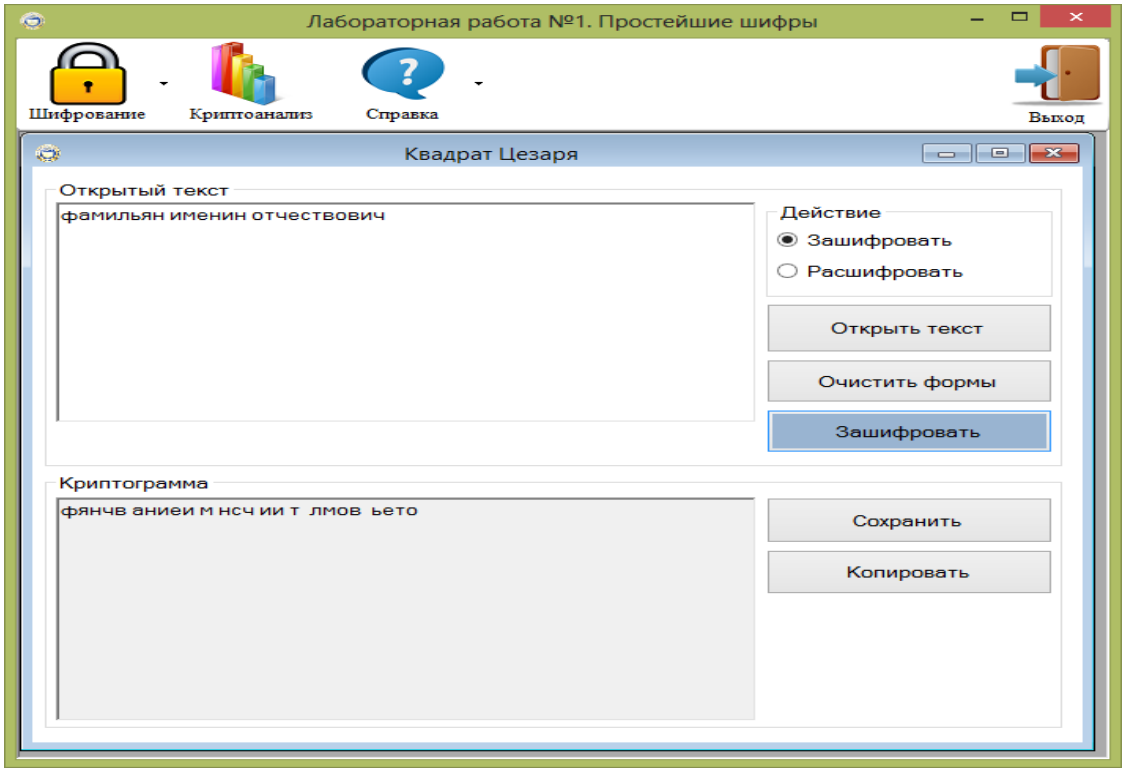


Рис. 1.24. **Зашифрований текст за допомогою квадрата Цезаря**

3. Скопіюйте визначену криптограму у форму відкритого тексту.

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.25).

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ. Якщо кількість символів не дорівнювала квадрату цілого числа, то цей текст буде містити кілька додаткових літер у кінці (див. рис. 1.24).

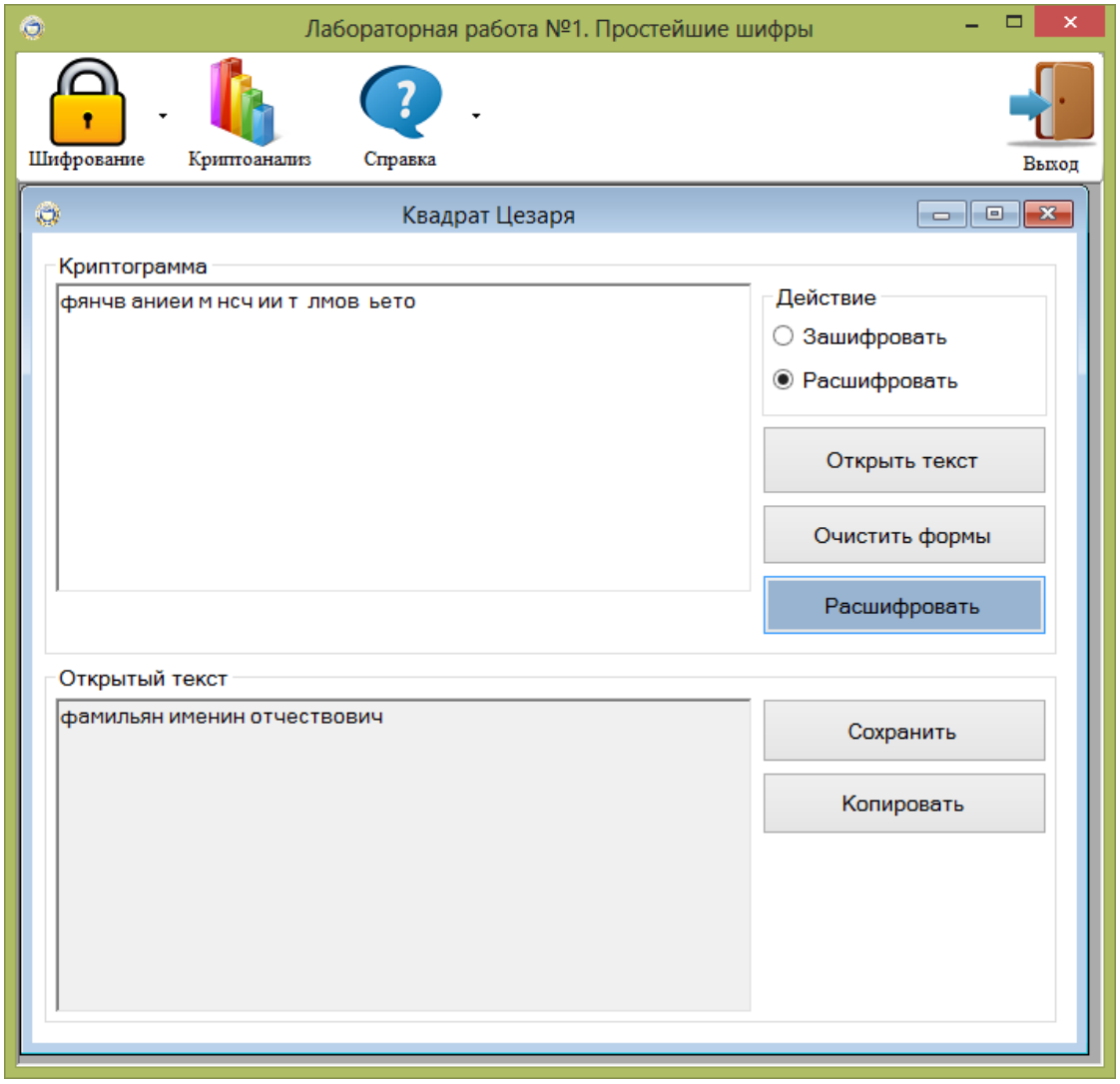


Рис. 1.25. **Розшифрування шифротексту**

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр Трітеміуса:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрити його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися тільки із цифр).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.26).

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.27).

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.26).

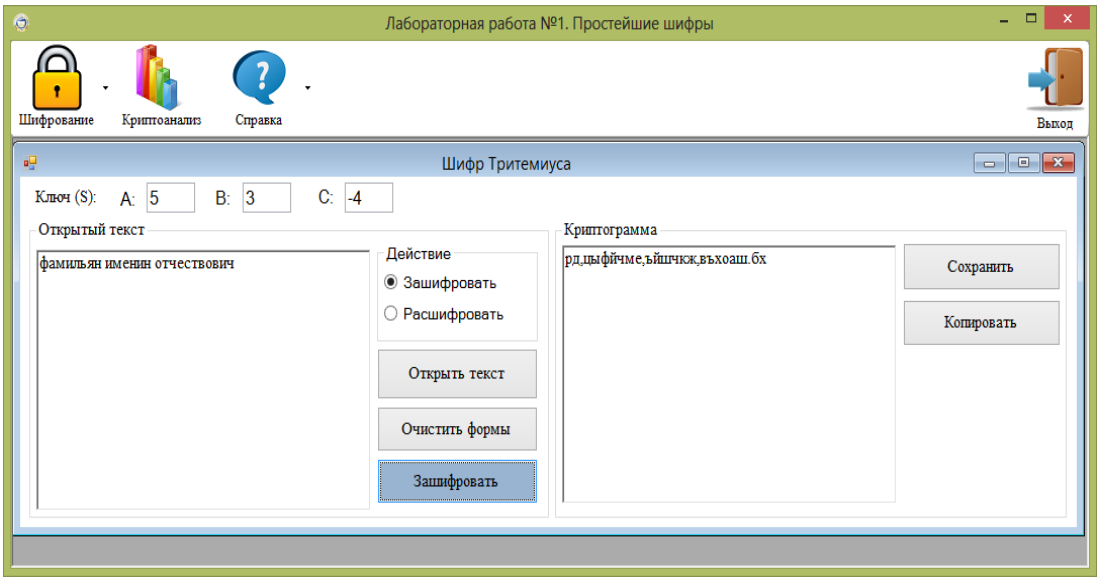


Рис. 1.26. **Зашифрований текст за допомогою шифру Трітеміуса**

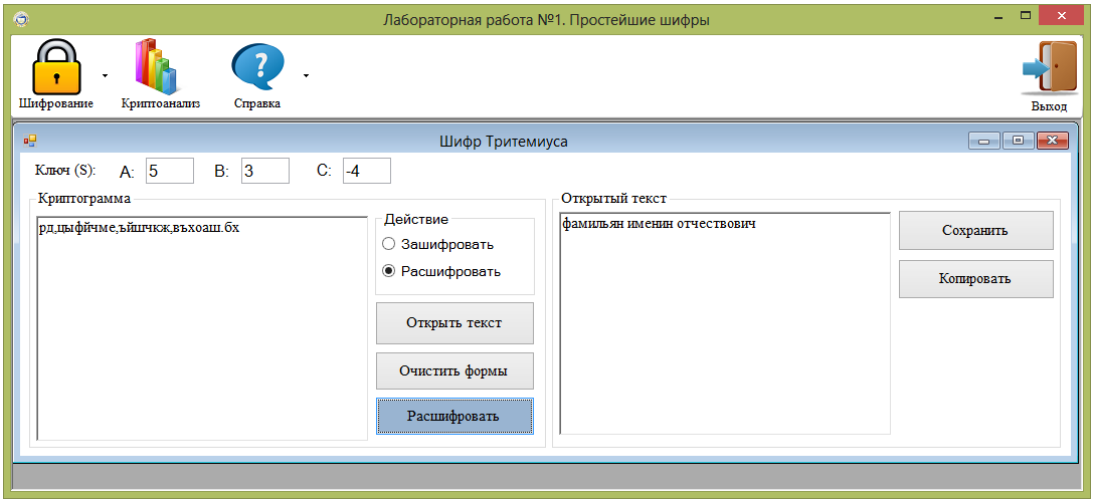


Рис. 1.27. **Розшифрування шифротексту**

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр Плейфера:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.28).

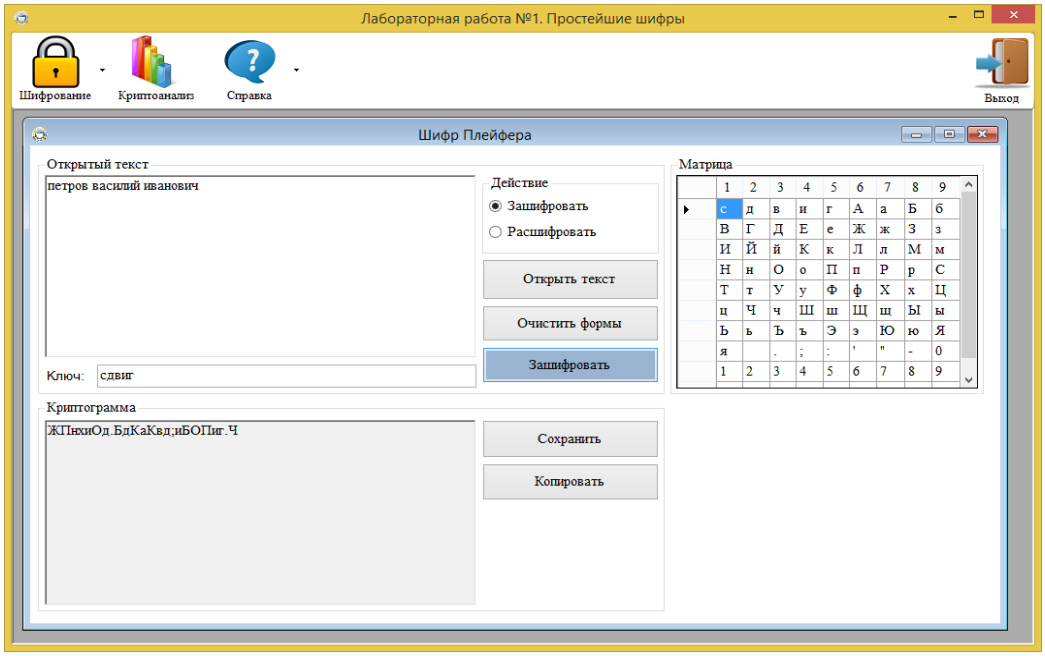


Рис. 1.28. **Зашифрований текст за допомогою шифру Плейфера**

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.29).

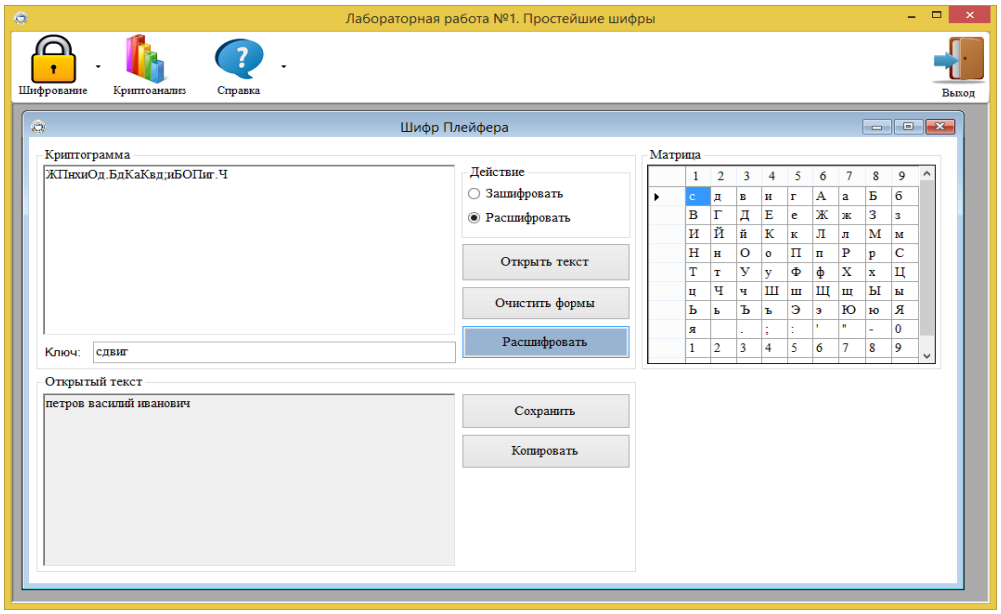


Рис. 1.29. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.28).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Дошка (квадрат) Полібія:

без використання ключа:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.30).

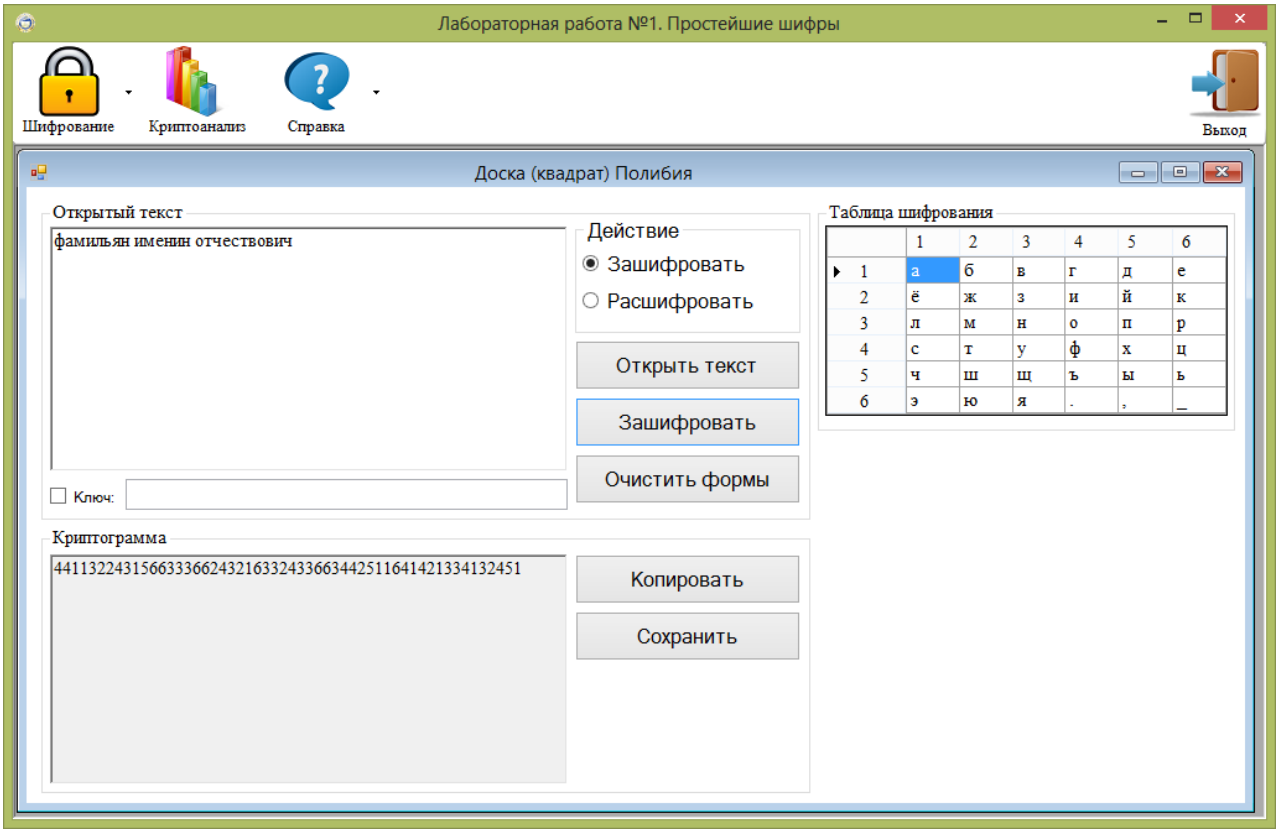


Рис. 1.30. **Зашифрований текст за допомогою квадрата Полібія без використання ключа**

3. Скопіюйте визначену криптограму у форму відкритого тексту.

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.31).

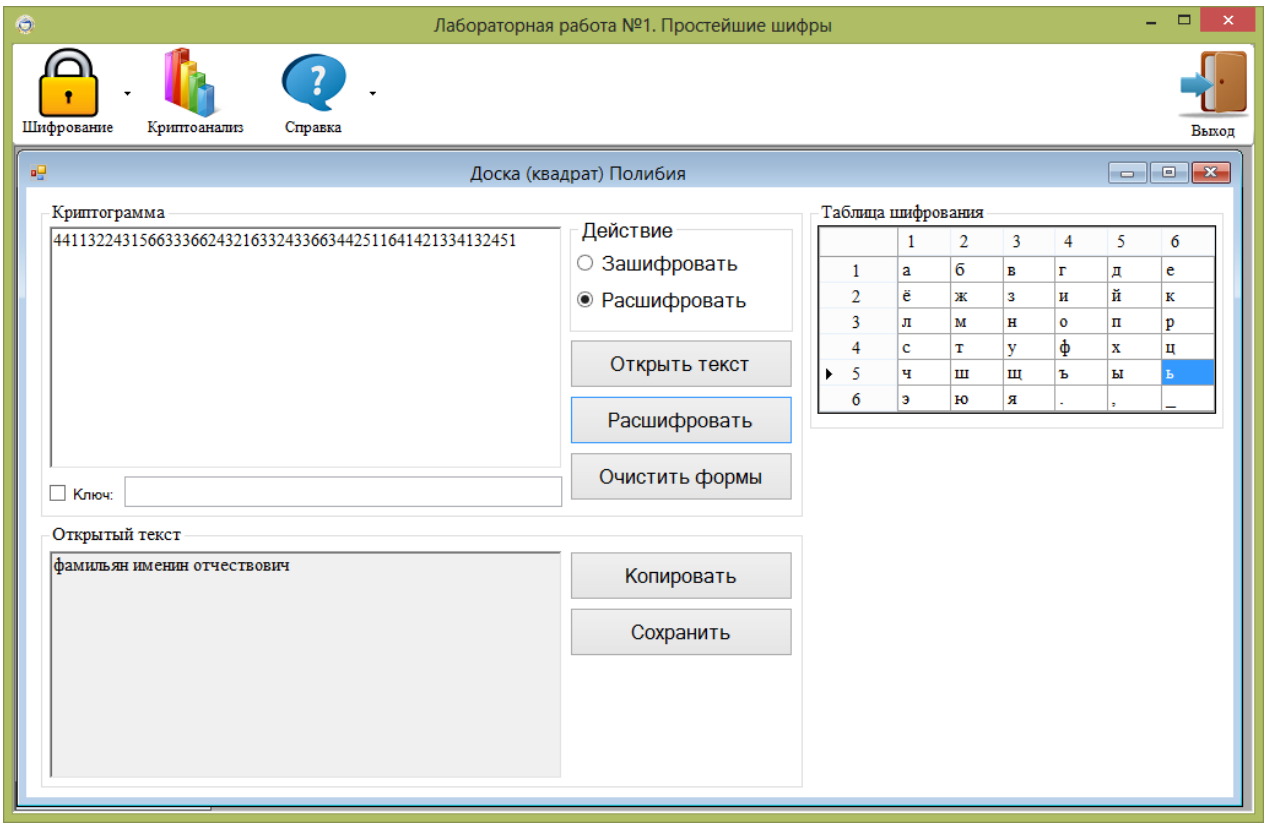


Рис. 1.31. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.30).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы";

із використанням ключа:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Поставте галочку та введіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.32).

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.33).

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.32).

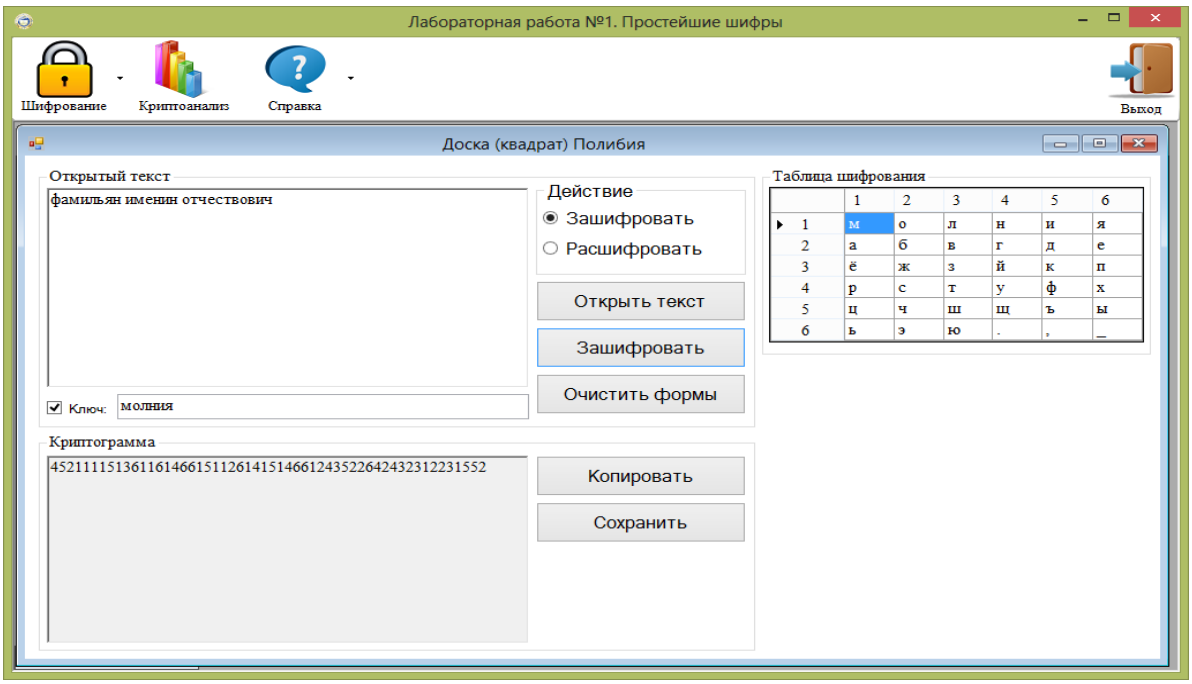


Рис. 1.32. **Зашифрований текст за допомогою квадрата Полібія з використанням ключа**

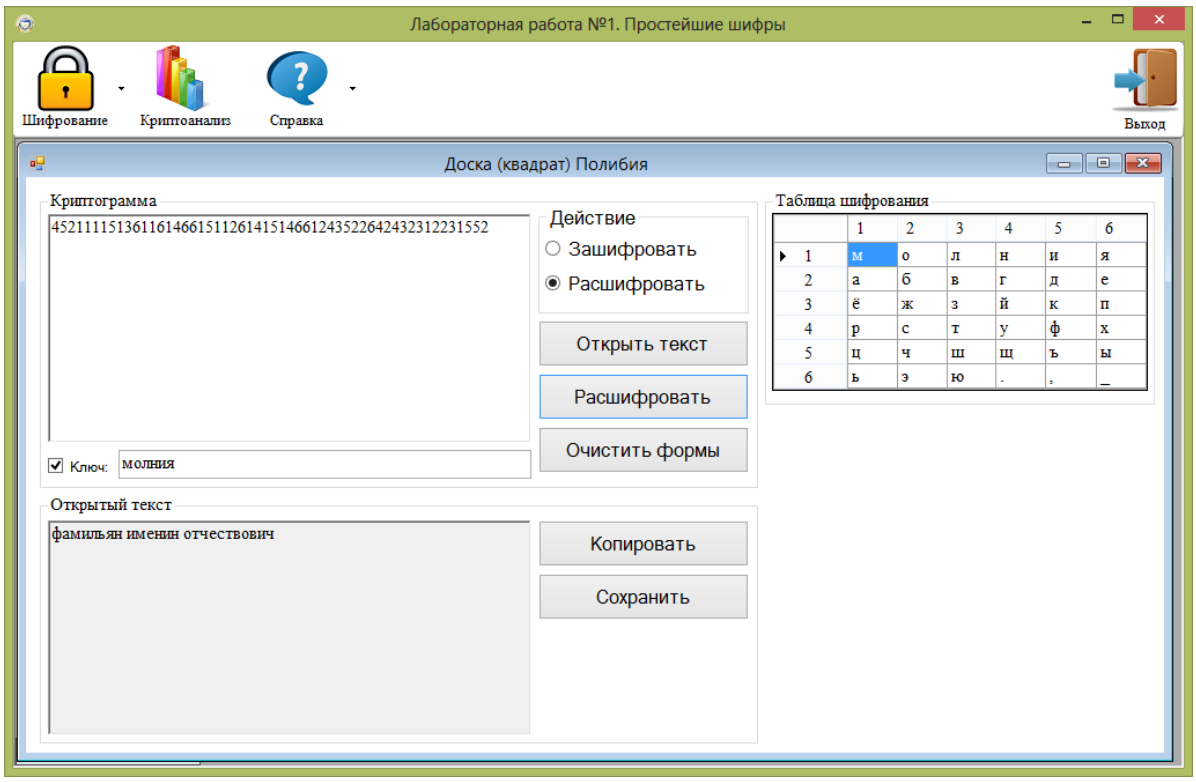


Рис. 1.33. **Розшифрування шифротексту**

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр Вітстона:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключі (вони мають складатися тільки із цифр).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.34).

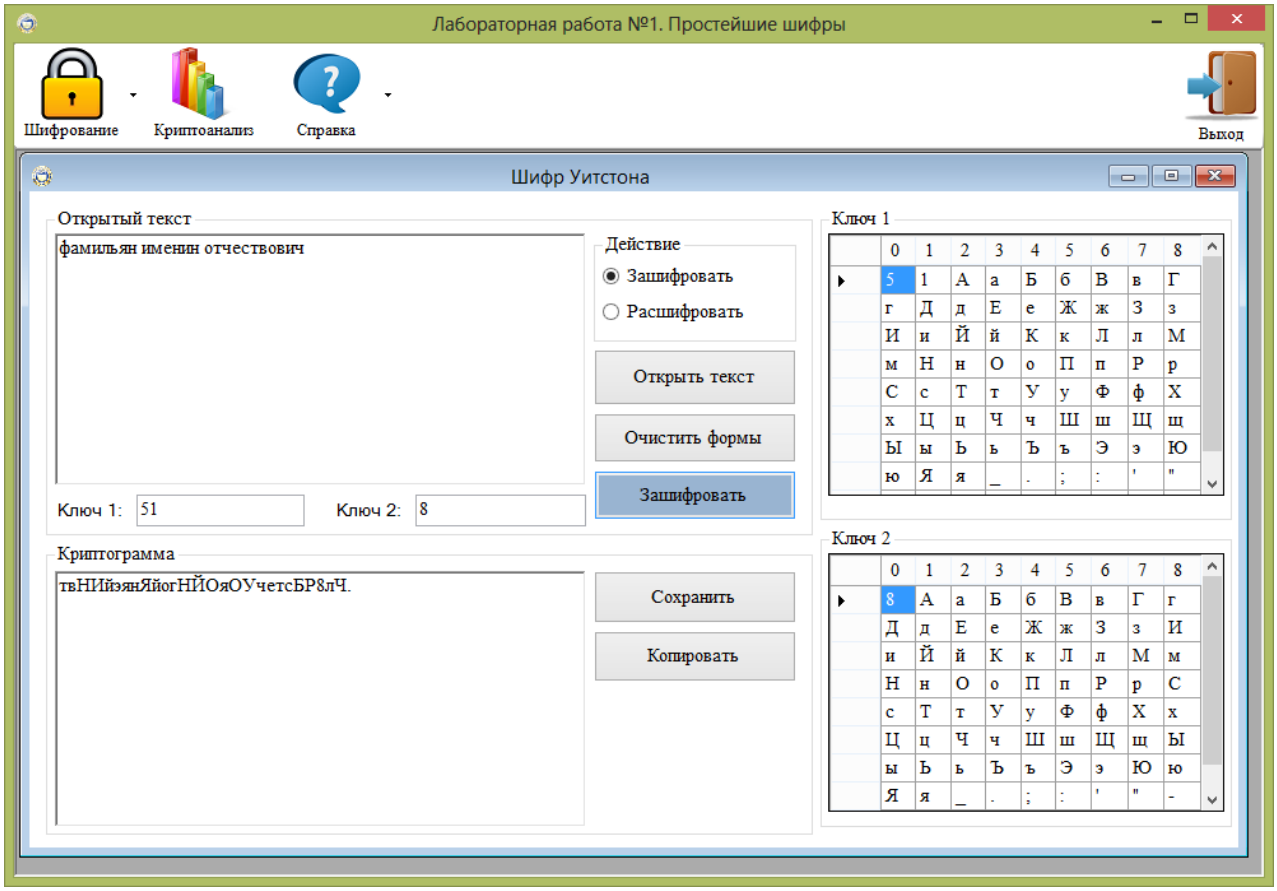


Рис. 1.34. **Зашифрований текст за допомогою шифру Вітстона**

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.35).

6. Визначена криптограма повинна бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.34).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

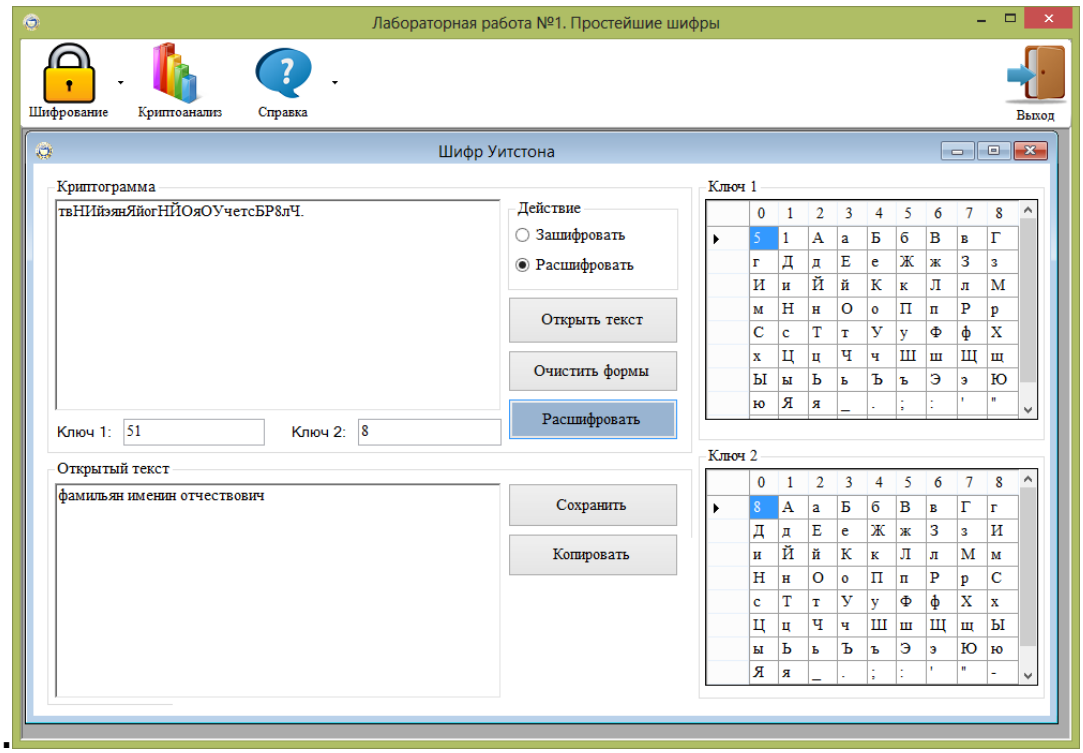


Рис. 1.35. **Розшифрування шифротексту**

Шифр Ґронсфельда:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися тільки із цифр).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.36).

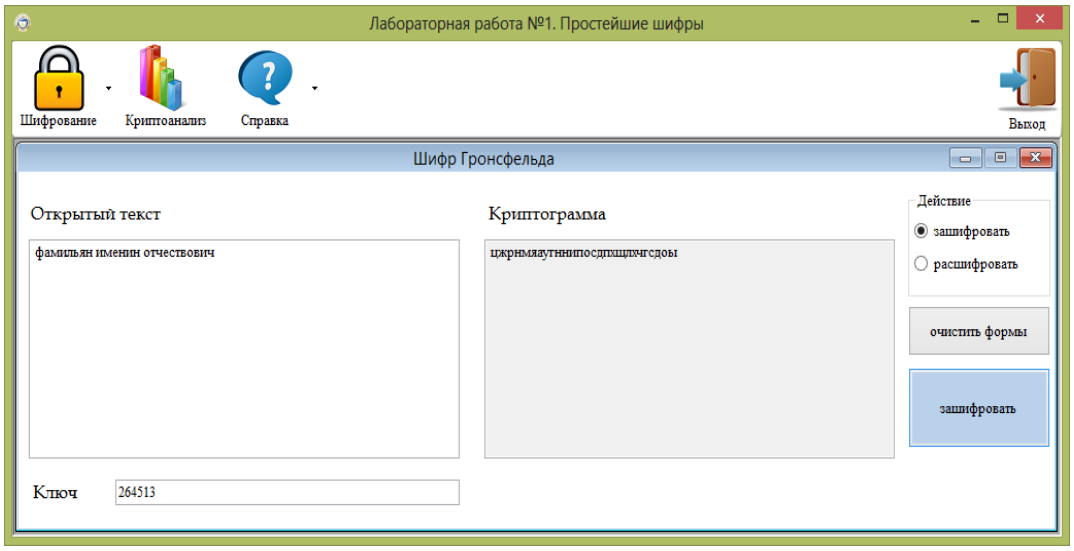


Рис. 1.36. **Зашифрований текст за допомогою шифру Ґронсфельда**

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.37).

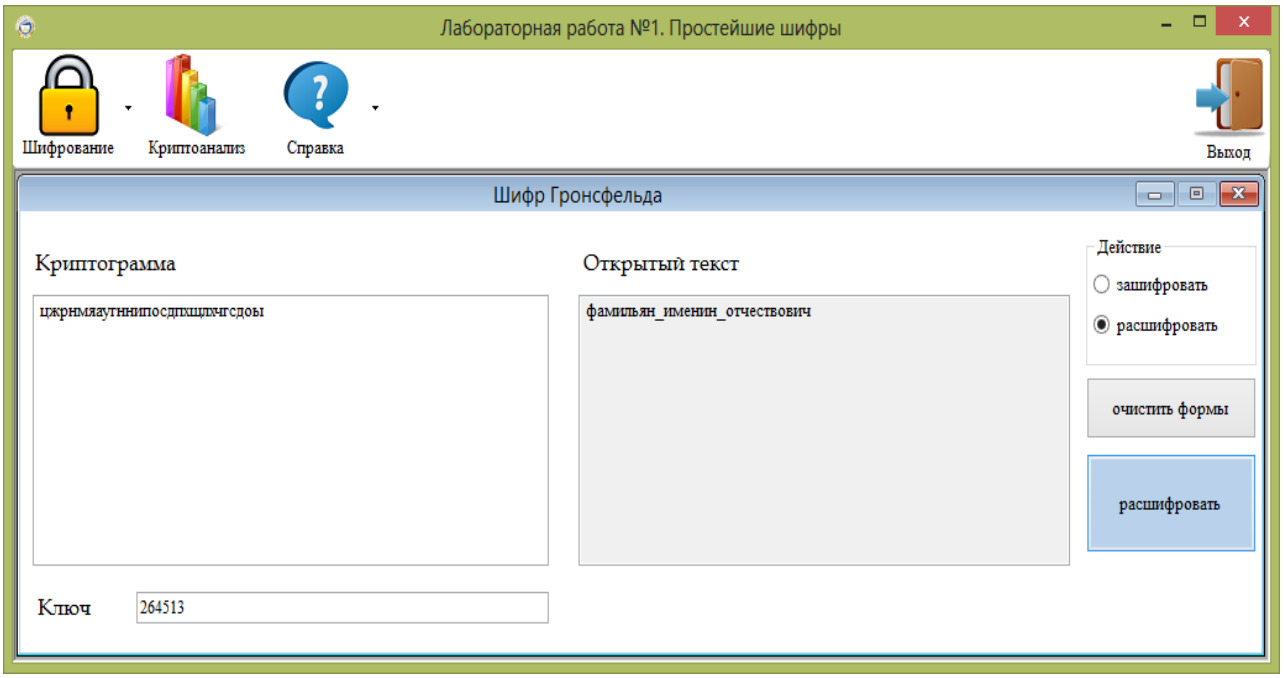


Рис. 1.37. **Розшифрування шифротексту**

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.36).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр Віженера:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Для перегляду таблиці Віженера необхідно натиснути кнопку "Открыть таблицу Віженера".

4. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.38).

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.39).

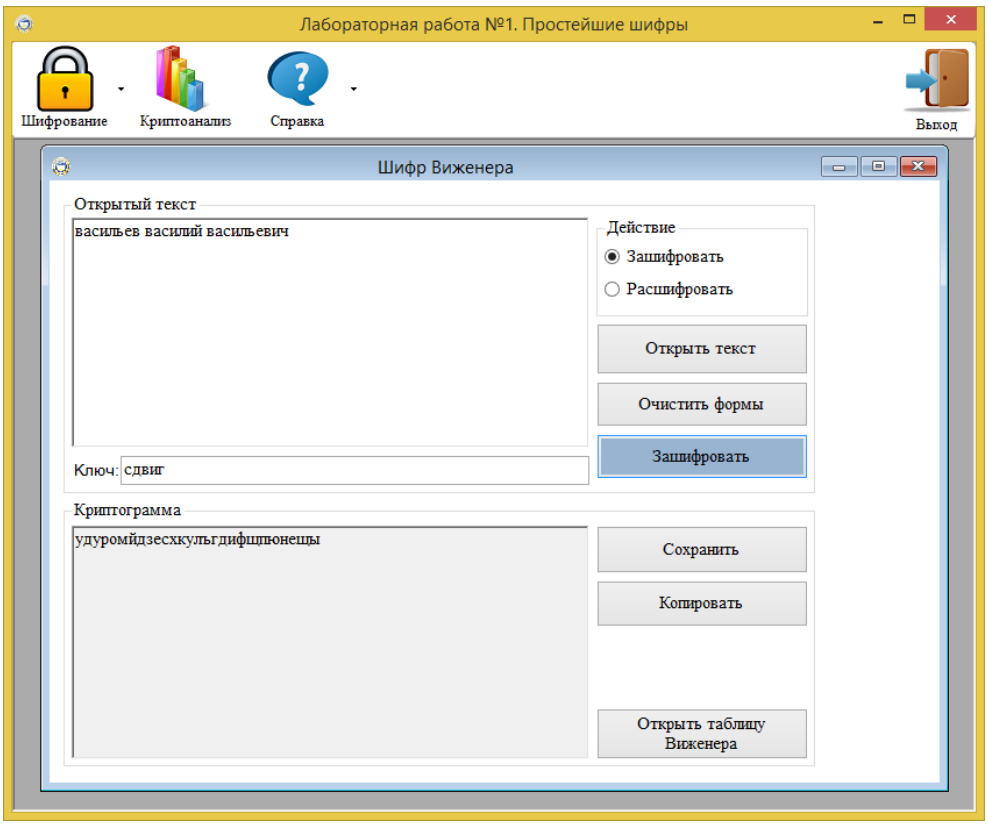


Рис. 1.38. **Зашифрований текст за допомогою шифру Віженера**

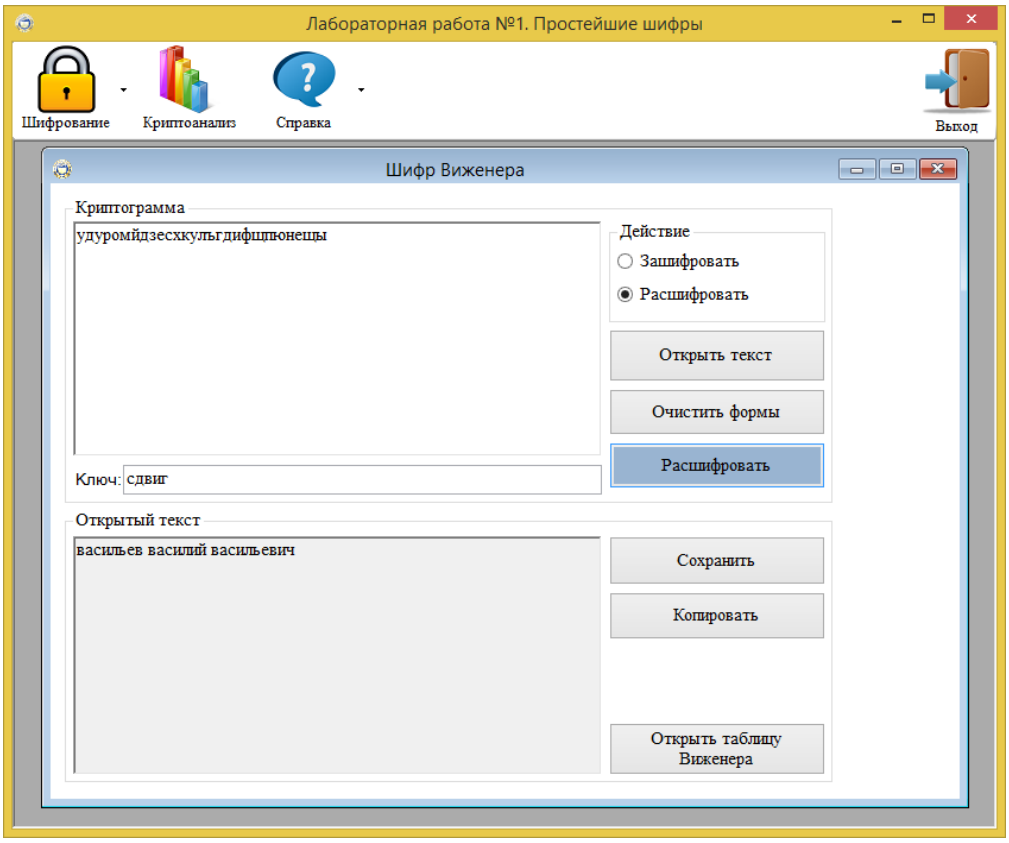


Рис. 1.39. **Розшифрування шифротексту**

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.38).

Для очищення всіх полів необхідно натиснути кнопку "Очистить формы".

Шифр Віженера з автоключем із використанням відкритого тексту:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Для перегляду таблиці Віженера необхідно натиснути кнопку "Открыть таблицу Виженера".

4. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.40).

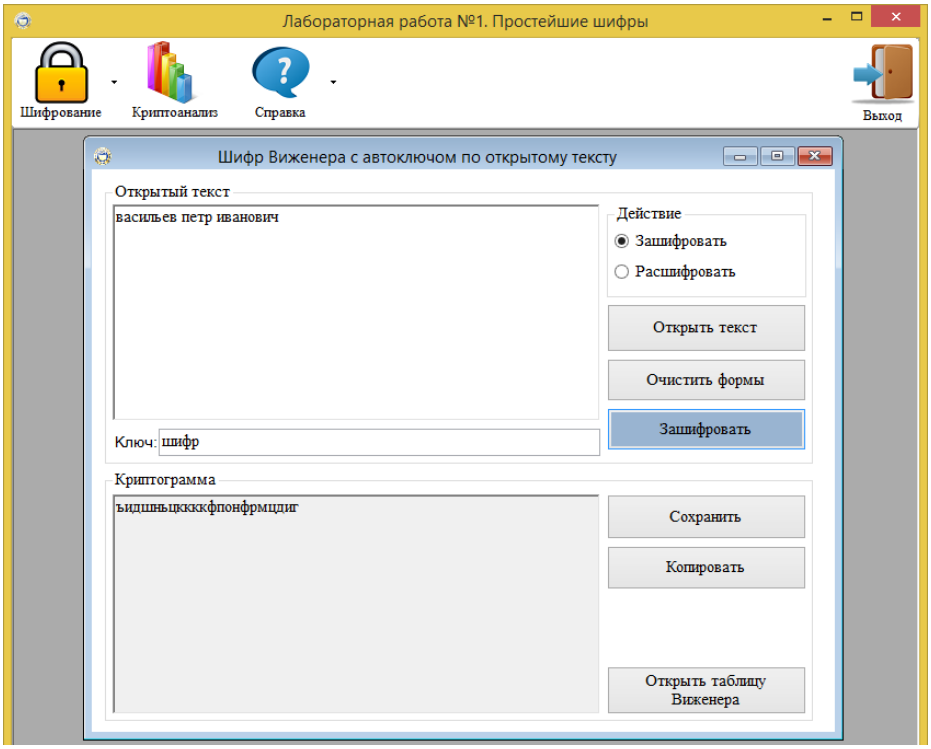


Рис. 1.40. **Зашифрований текст за допомогою шифру Віженера з автоключем із використанням відкритого тексту**

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.41).

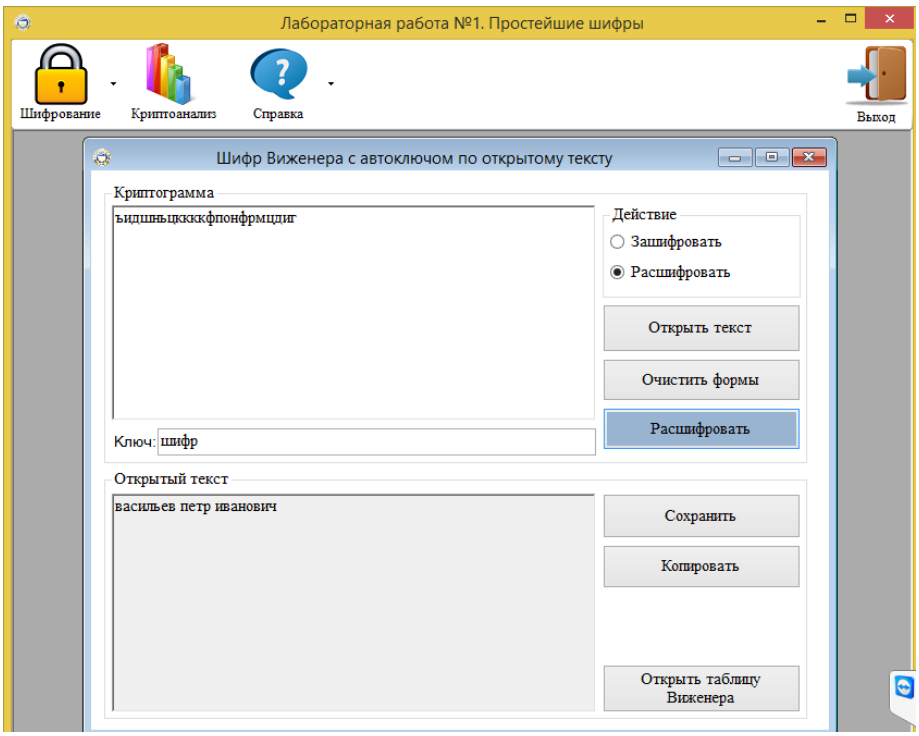


Рис. 1.41. **Розшифрування шифротексту**

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.40).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Шифр з автоключем із використанням криптограми:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Уведіть ключ (він має складатися з малих літер російського алфавіту, також може мати пробіли).

3. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.42).

4. Скопіюйте визначену криптограму у форму відкритого тексту.

5. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.43).

6. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.42).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

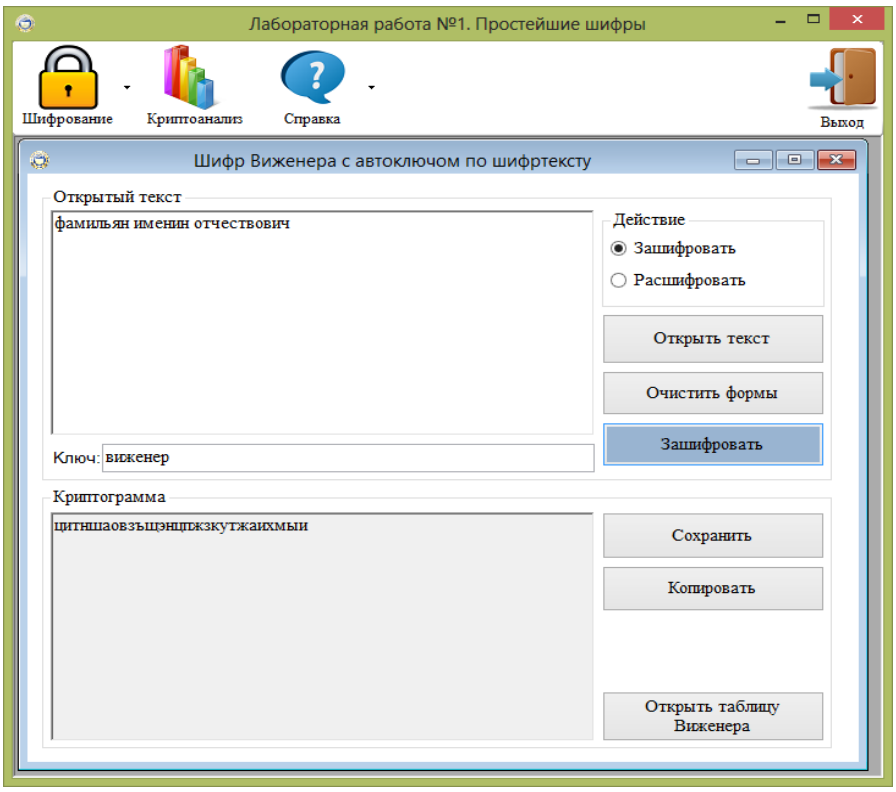


Рис. 1.42. **Зашифрований текст за допомогою шифру Віженера з автоключем із використанням криптограми**

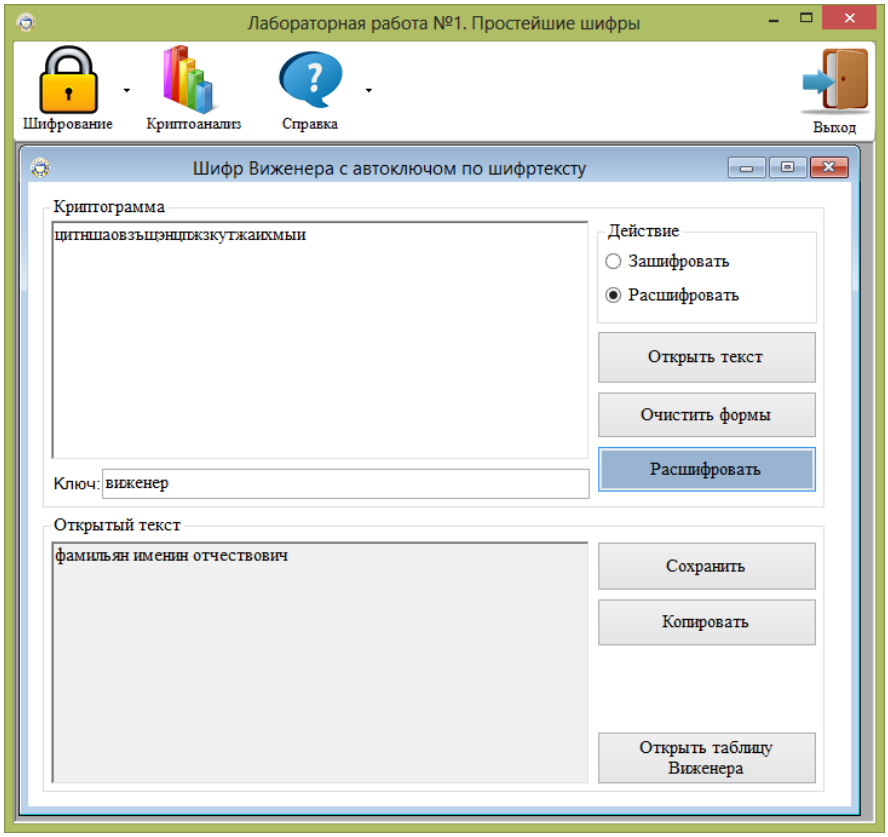


Рис. 1.43. **Розшифрування шифротексту**

Поліалфавітна заміна:

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Натисніть кнопку "Заполнить таблицу функциями по умолчанию" або додайте свою функцію, заповнивши необхідні поля.

3. Уведіть ключ (він має складатися тільки із цифр, значення яких не більше, ніж кількість вибраних функцій; рахунок починають із 0).

4. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.44).

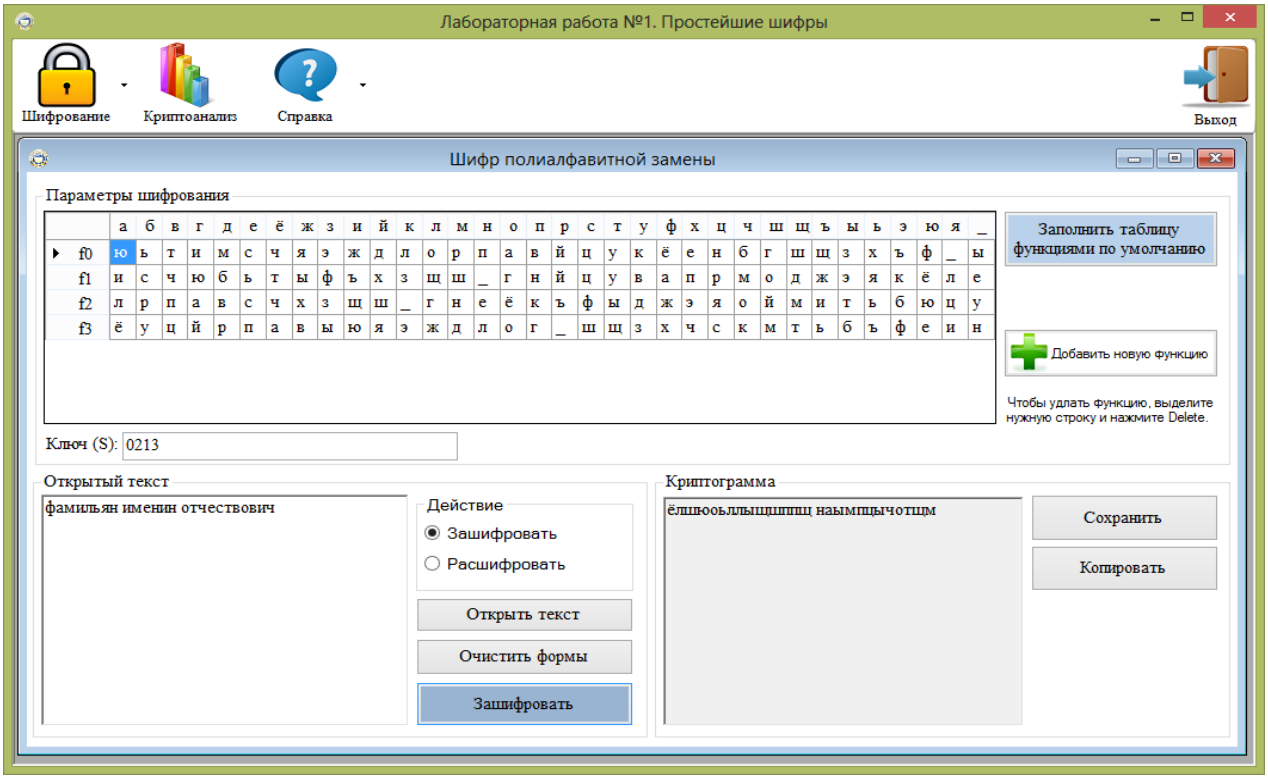


Рис. 1.44. **Зашифрований текст за допомогою поліалфавітної заміни**

5. Скопіюйте визначену криптограму у форму відкритого тексту.

6. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.45).

7. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.44).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

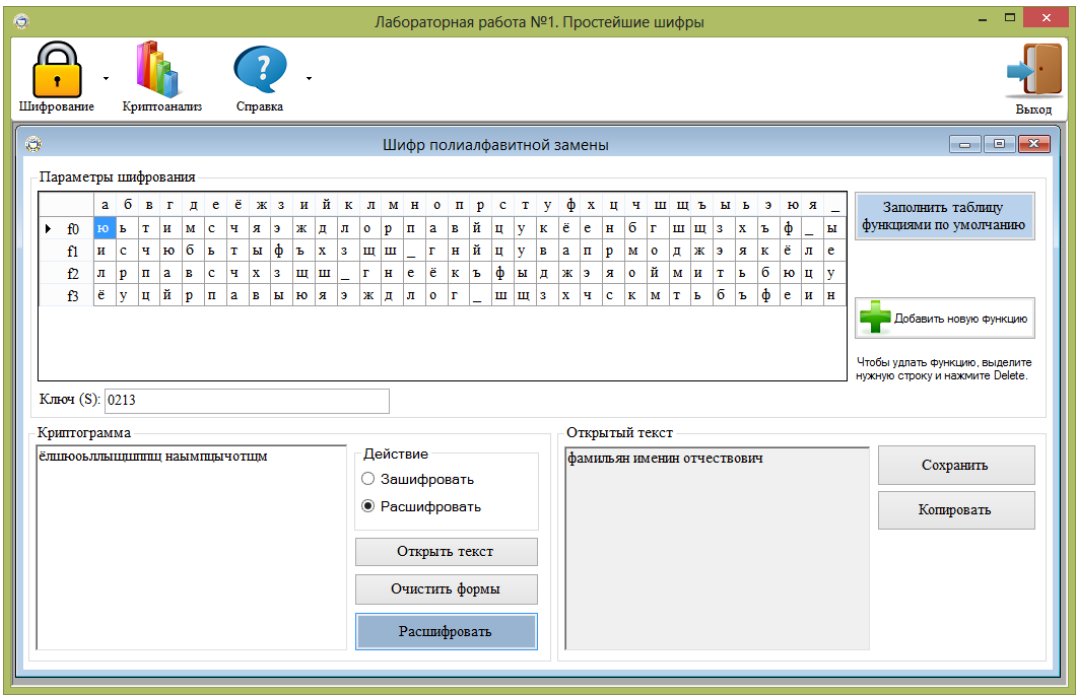


Рис. 1.45. **Розшифрування шифротексту**

Шифрувальної машина "Енігма":

1. Уведіть ПІБ як відкритий текст (він має складатися з малих літер російського алфавіту, також може містити пробіли) або відкрийте його з файлу, натиснувши кнопку "Открыть текст".

2. Виберіть дію "Зашифровать" і натисніть кнопку "Зашифровать" (рис. 1.46).

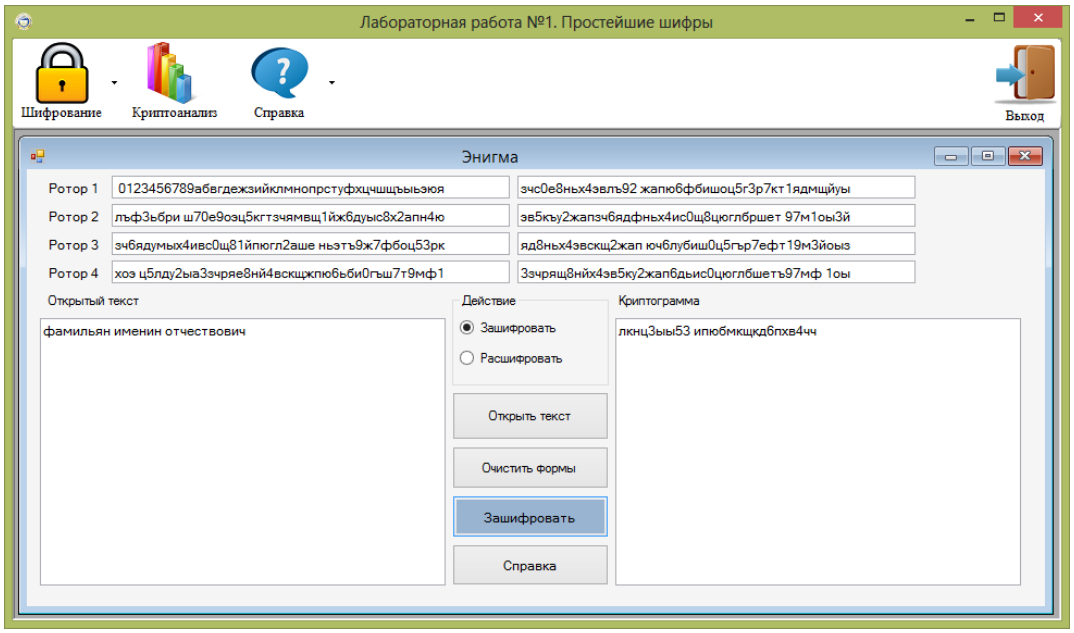


Рис. 1.46. **Зашифрований текст за допомогою шифрувальної машини "Енігма"**

3. Скопіюйте визначену криптограму у форму відкритого тексту.

4. Виберіть дію "Расшифровать" і натисніть кнопку "Расшифровать" (рис. 1.47).

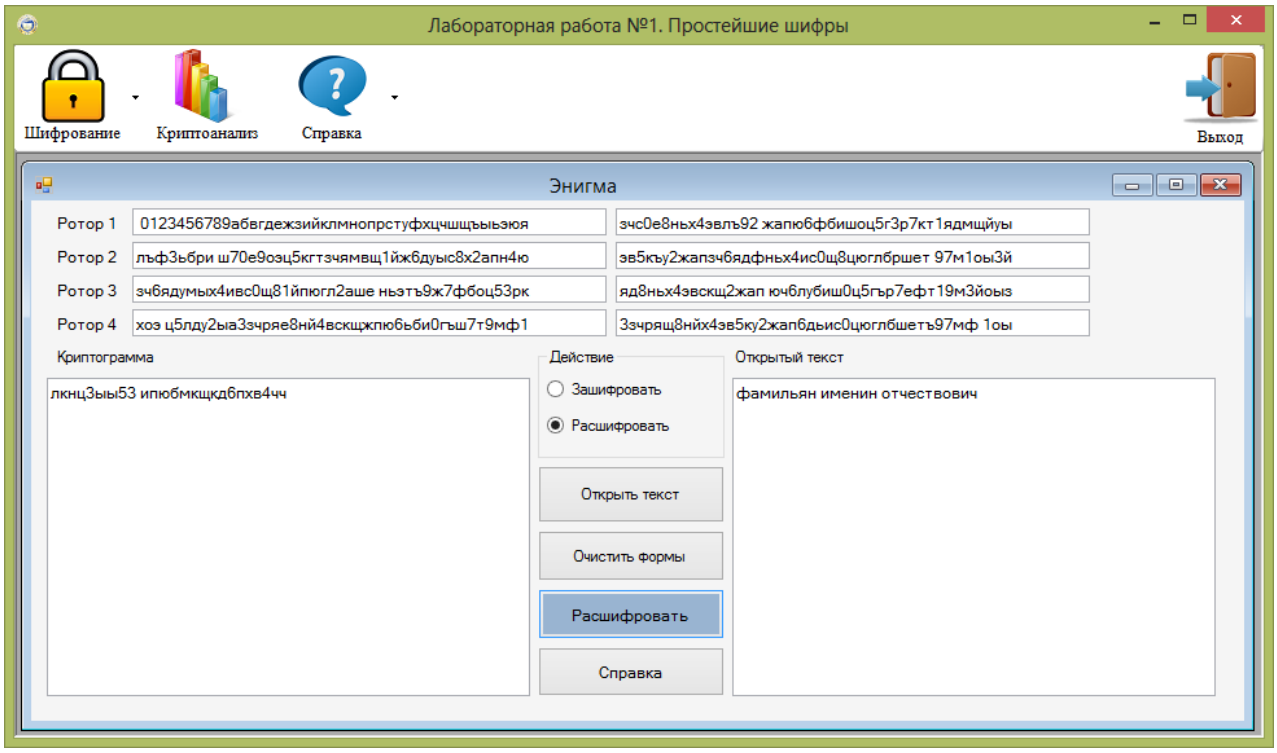


Рис. 1.47. **Розшифрування шифротексту**

5. Визначена криптограма має бути такою самою, як і введений відкритий текст ПІБ (див. рис. 1.46).

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить формы".

Частотний аналіз:

1. Зашифруйте текст великого розміру будь-яким алгоритмом (наприклад, шифром Цезаря).

2. Визначений зашифрований текст скопіюйте в поле "Криптограма" частотного криптоаналізу.

3. Натисніть кнопку "Рассчитать частоту" (рис. 1.48).

4. Натисніть кнопку "Заменить все".

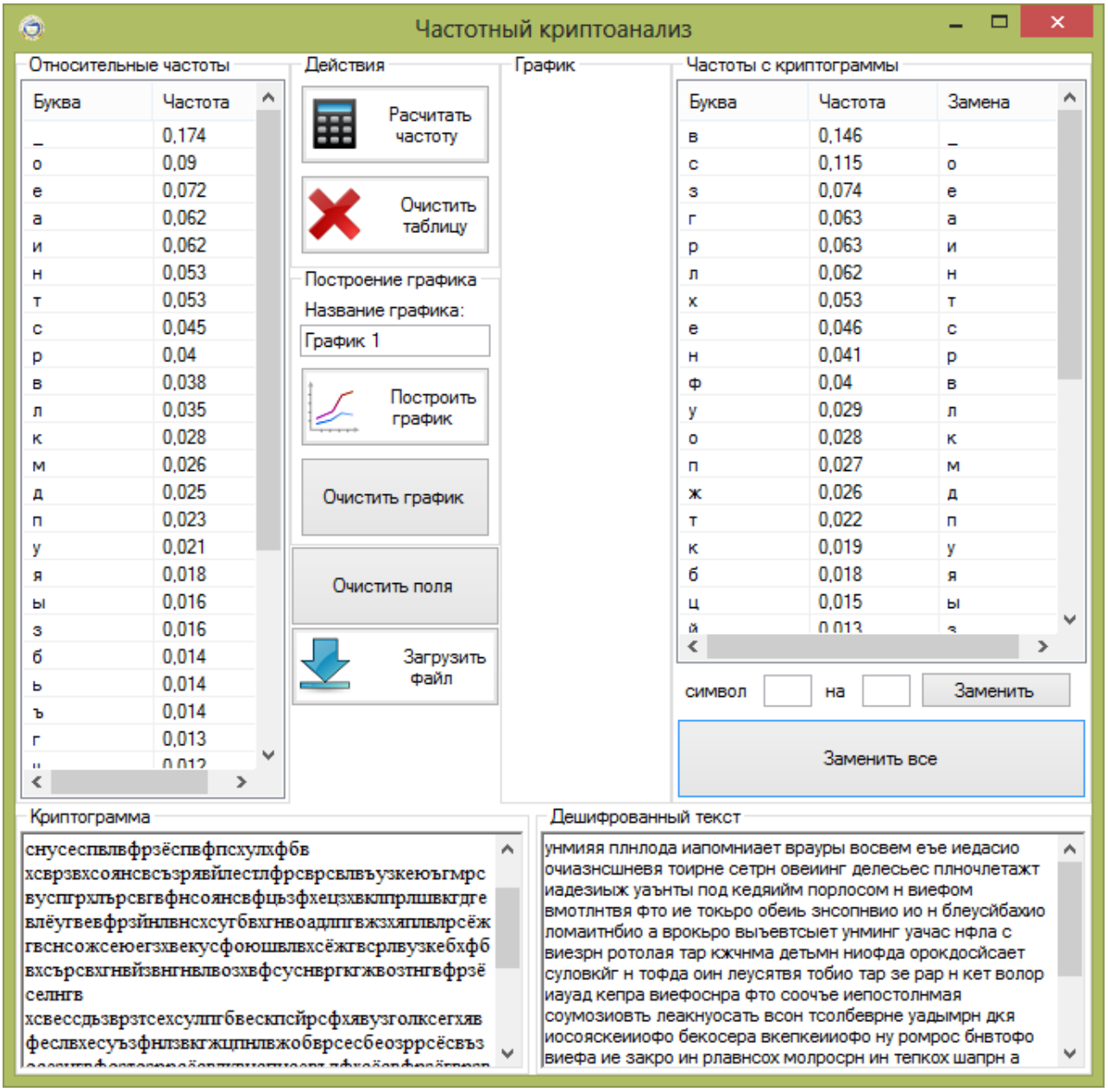


Рис. 1.48. **Частотний криптоаналіз**

Якщо дешифрування тексту містить помилки, то деякі букви можна підібрати та замінити за змістом, натиснувши кнопку "Заменить".

5. Збудуйте графік частоти входження букв у тексті, натиснувши кнопку "Построить график" (рис. 1.49).

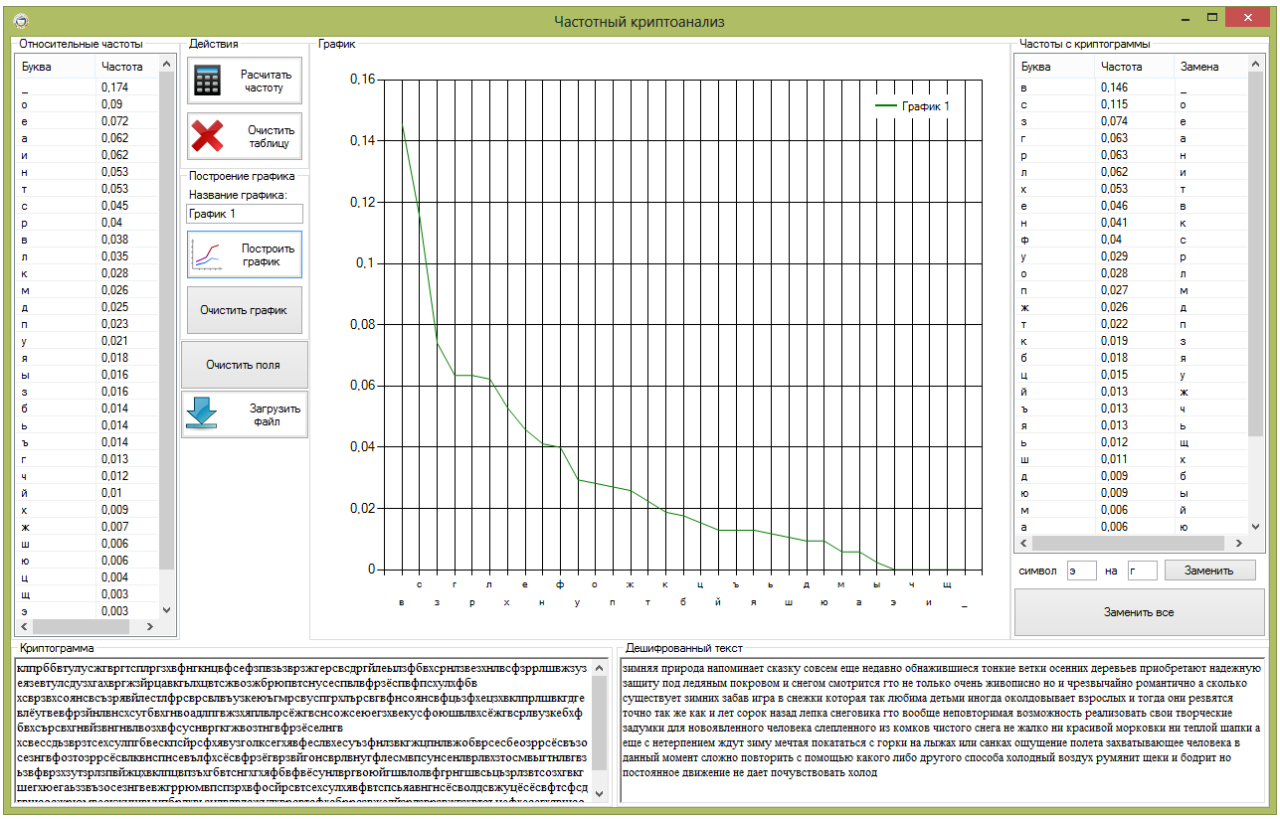


Рис. 1.49. **Побудований графік частоти входження букв у текст**

Для того щоб очистити всі поля, необхідно натиснути кнопку "Очистить поля".

1.5. Завдання до лабораторної роботи

1. Виконайте шифрування свого прізвища, імені та по батькові за всіма розглянутими в п. 1.3 методами шифрування.

2. Здійсніть розшифрування свого прізвища, імені та по батькові за всіма розглянутими в п.1.3 методами шифрування.

3. У звіт включіть опис усіх дій, як під час шифрування, так і розшифрування.

4. Оцініть криптографічну стійкість шифрів на основі порівняння множини ключів (кількості) *K* і множини криптограм *C*. Завдання виконайте, відповідно до варіанта (табл. 1.6).

5. Порівняйте статистичну залежність криптограм і відкритого тексту.

6. Додаткове завдання: створіть програму, яка виконує шифрування та розшифрування тексту за методами, відповідно до варіанта (див. табл. 1.5).

Таблиця 1.6

Варіанти індивідуального завдання

Варіанти	Методи шифрування
1	Шифр Плейфера. Поліалфавітна заміна
2	Перестановний шифр із ключовим словом. Шифр з автоключем із використанням криптограми
3	Шифр простої заміни. Шифр з автоключем із використанням відкритого тексту
4	Афінна криптосистема. Поліалфавітна заміна
5	Шифр Цезаря. Шифр Віженера
6	Шифр Цезаря із ключовим словом. Шифр з автоключем із використанням відкритого тексту
7	Матрична перестановка. Поліалфавітна заміна
8	Шифр Плейфера. Шифр з автоключем із використанням криптограми
9	Перестановний шифр із ключовим словом. Поліалфавітна заміна
10	Шифр простої заміни. Шифр Віженера
11	Афінна криптосистема. Шифр з автоключем із використанням відкритого тексту
12	Шифр Цезаря. Шифр з автоключем із використанням криптограми

1.6. Контрольні запитання

1. Розкрийте сутність процесу шифрування та шифру.
2. Що таке "конфіденційна інформація"?
3. Дайте визначення поняття "конфіденційність".
4. Розкрийте сутність ролі криптоаналітика.
5. Як можна визначити криптографічну стійкість за криптограмою для простих шифрів?
6. Розкрийте сутність частотного криптоаналізу простих шифрів.
7. Як використовують перестановку у простих шифрах?
8. Яким чином використовують підстановку у простих шифрах?
9. Яка є класифікація простих шифрів? Дайте характеристику кожного із класів.
10. Які характеристики повинен мати шифр, щоб протистояти частотному криптоаналізу?

Лабораторна робота 2
Блокові симетричні шифри

2.1. Мета

Мета цієї лабораторної роботи – набути теоретичні та практичні навички із забезпечення конфіденційності та цілісності інформації за допомогою блокових симетричних шифрів; ознайомитися з архітектурою таких шифрів та їхніми параметрами; набути практичні навички у використанні різноманітних режимів шифрування.

2.2. Рекомендації до підготовки до виконання

Необхідно ознайомитися з основами симетричного шифрування, принципами побудови симетричних блокових шифрів, перевагами та недоліками режимів використання таких шифрів.

2.3. Загальні теоретичні положення

Режими шифрування. Під *режимом шифрування* слід розуміти такий алгоритм застосування блокового шифру, що дозволяє перетворювати відкритий текст, довжина якого перевищує один блок, на шифротекст; після передавання шифротексту відкритим каналом відновити відкритий текст, підвищити стійкість до видів атак, реалізувати додаткові послуги безпеки.

За такого використання власне блоковий шифр є складовою частиною алгоритму застосування, оскільки він працює лише з окремим блоком даних, а використовують його для шифрування цілого повідомлення, яке складають з деякого числа n блоків. Більш того, довжина повідомлення може не бути кратною довжині блока, тобто його неможливо розподілити на ціле число таких блоків:

Режими шифрування:

ECB (Electronic Code Book) – електронна кодова книга;

CBC (Cipher Block Chaining) – зчеплення блоків шифротексту;

CFB (Cipher Feedback) – зворотне завантаження шифротексту;

OFB (Output Feedback) – зворотне завантаження вихідних даних.

Електронна кодова книга – ECB (Electronic Code Book). У цьому режимі шифрування/дешифрування *i*-го блока відкритого тексту/шифротексту виконують, незалежно від інших блоків:

$$c_i = E_k(m_i), \;\; m_i = D_k(c_i). \qquad (2.1)$$

Недоліком цього режиму шифрування є те, що однакові блоки вхідного тексту будуть перетворюватися на однакові блоки шифротексту, що дає можливість зловмисникові, по-перше, робити припущення про характер інформації у відкритому тексті, а по-друге, підмінити один або кілька блоків шифротексту. Перевагою режиму можна назвати простоту реалізації, а також можливість розпаралелювання процедури шифрування (рис. 2.1).

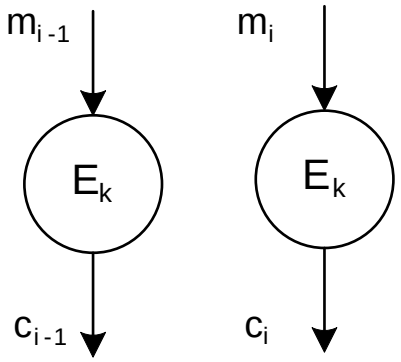


Рис. 2.1. **Можливість розпаралелювання режиму**

На вхід алгоритму надходить *k*-бітовий ключ *K* і *n*-бітові блоки відкритого тексту $P = p_1, p_2, \ldots, p_i, \ldots$. На виході формуються *n*-бітові блоки шифротексту $C = c_1, c_2, \ldots, c_i, \ldots$. (рис. 2.2).

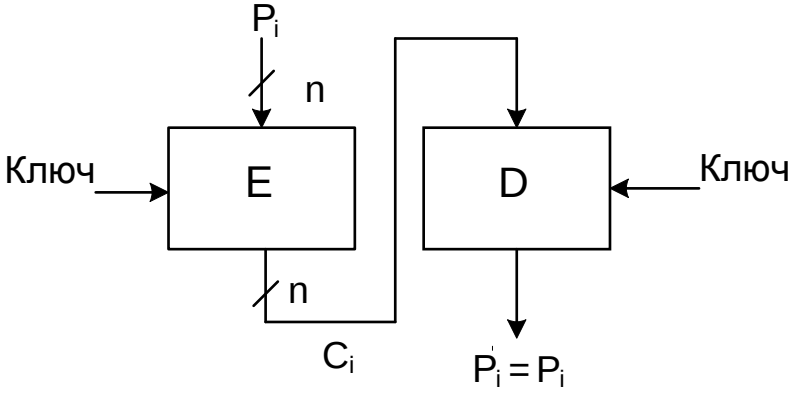


Рис. 2.2. **Режим електронної кодової книги**

Шифрування: $c_i = E_k(p_i)$.

Дешифрування: $p_i = D_k(c_i)$.

Слід зазначити такі властивості ECB-режиму:

1) блокові шифри в такому режимі не забезпечують приховання шаблонів даних – ідентичні блоки відкритого тексту (за того самого ключа) перетворяться на ідентичний шифротекст;

2) блоки шифрують, незалежно один від одного. Перевпорядкування блоків шифротексту призведе до перевпорядкування блоків відкритого тексту. Оскільки блоки шифротексту незалежні, навмисна підстановка ECB-блоків (наприклад, вставка блока, що часто зустрічається) не впливає на дешифрування суміжних блоків;

3) один і більше помилкових бітів в одному блоці шифротексту впливають на розшифрування тільки цього блока. Результат дешифрування блока з помилками є випадковою величиною, можна відновити близько 50 % бітів відкритого тексту;

4) істотним недоліком ECB є те, що однакові блоки відкритого тексту на однаковому ключі завжди мають однаковий зашифрований вигляд. Тому для великих повідомлень ECB-режим уважають небезпечним. Якщо повідомлення має багато однакових блоків, то під час криптоаналізу цю закономірність буде виявлено.

Зазначені особливості обмежують використання режиму ECB: його не рекомендують застосовувати для шифрування повідомлень, довжиною більшою за один блок, або якщо повторно використовують ключі для шифрування більше ніж одного одноблокового повідомлення. У цьому разі секретність ключа можна забезпечити додаванням до кожного блока випадкових бітів.

Зчеплення блоків шифротексту – CBC (Cipher Block Chaining). Режим CBC передбачає такі алгоритми шифрування/розшифрування:

$$c_i = E_k\left(m_i \oplus c_{i-1}\right),\ \ m_i = D_k(c_i) \oplus c_{i-1}. \qquad (2.2)$$

У режимі CBC кожний блок відкритого тексту додають до блоків шифротексту, визначених на попередньому етапі. Таким чином, відбувається зчеплення блоків один з одним. Незалежна маніпуляція з кожним із них неможлива, а однакові вхідні блоки будуть давати на виході різні результати. Водночас ускладнюється розпаралелювання процедури шифрування (рис. 2.3).

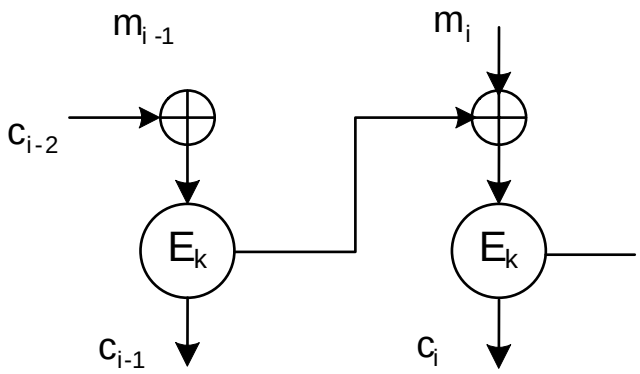


Рис. 2.3. **Ускладнена процедура розпаралелювання**

Режим зчеплення блоків шифру містить використання *n*-розрядного вектора ініціалізації *IV* (*Initializing Vector*). У схему додають регістр або буфер зворотного зв'язку, куди спочатку записують значення *IV*, а потім – наступні значення блоків шифротексту. На рис. 2.4. подано схему CBC. Для визначення першого блока зашифрованого повідомлення використовують вектор ініціалізації (*IV*), для якого виконують операцію *XOR* із першим блоком незашифрованого повідомлення. Під час розшифрування для *IV* виконують операцію *XOR* із виходом алгоритму розшифрування для створення першого блока незашифрованого тексту.

IV має бути відомий як відправнику, так і отримувачу. Для максимальної безпеки він має бути захищений так само, як і ключ.

На вхід алгоритму надходить *k*-бітовий ключ *K*, *n*-бітовий *IV*, послідовність *n*-бітових блоків відкритого тексту $P = p_1, p_2, p_3, ..., p_i, p_t$. (рис. 2.4).

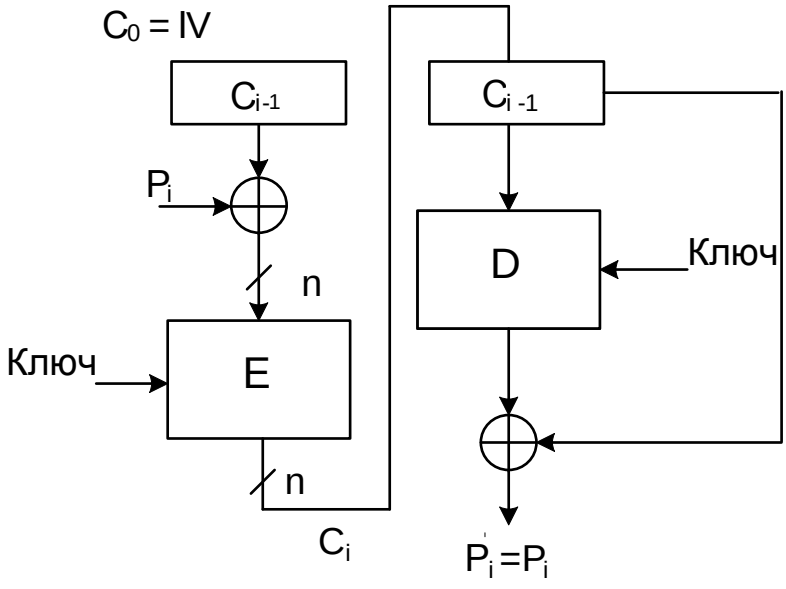


Рис. 2.4. **Режим зчеплення блоків шифру**

Послідовність блоків шифротексту $C = c_1, c_2, \ldots, c_t$ обчислюють за таким правилом: $c_i = E_k(p_i \oplus c_{i-1})$; $i = 1, \ldots, t$; $c_0 = IV$.

Розшифрування здійснюють так: $p_i = c_{i-1} \oplus D_k(c_i)$;

$i = 1, \ldots, t$; $c_0 = IV$.

Властивості CBC-режиму:

1) ідентичні блоки шифротексту формують тоді, коли шифрують той самий відкритий текст на тих самих ключах і тому самому векторі ініціалізації. Зміна вектора ініціалізації, ключа або першого блока повідомлення (наприклад, із використанням лічильника або випадкової величини) призводить до різних шифротекстів;

2) механізм зчеплення призводить до того, що шифротекст c_i залежить від p_i і всіх попередніх блоків відкритого тексту. У цьому разі залежність від попередніх блоків полягає у значенні попереднього блока шифротексту. Отже, зміна порядку проходження блоків шифротексту впливає на розшифрування. Для того щоб правильно розшифрувати блок шифротексту, необхідно мати правильний попередній блок шифротексту;

3) одинична бітова помилка у блоці шифротексту c_i впливає на дешифрування як мінімум двох блоків c_i і c_{i+1} (оскільки p_i залежить від c_i і c_{i+1}). Блок p'_i відновлений із c_i, переважно, є повністю випадковим (50 % бітів помилкові), тоді як відновлений відкритий текст p'_{i+1} буде мати неправильних бітів саме стільки, скільки їх має блок c_i. Таким чином, зловмисник може викликати передбачувані зміни бітів у p'_{i+1}, змінюючи відповідні біти блока c_i;

4) CBC-режим є реалізацією шифрування з автоключем за шифротекстом у тому розумінні, що якщо є помилка у блоці c_i (включаючи втрату одного або більше вхідних блоків), але немає помилок у c_{i+1}, c_{i+2}, то можливе коректне дешифрування p_{i+2}.

Із погляду практичного використання блокових шифрів у цьому режимі необхідно також ураховувати такі рекомендації:

1) *поширення помилок під час шифрування*. Хоча розшифрування у CBC-режимі дозволяє відновити відкритий текст із помилок у блоках шифротексту, модифікація блока відкритого тексту p_i під час шифрування змінює всі наступні блоки шифротексту. Це ускладнює (якщо не виключає) використання режимів зчеплення, коли необхідний випадковий доступ (читання/запису) до зашифрованих даних. Альтернативою є режим ECB;

2) *самосинхронізація та групування помилок.* Незважаючи на те що самосинхронізація можлива за наявності помилок у бітах, за "загублених" бітів, що є причиною помилки визначення меж блока (помилки цілісності кадру), вона неможлива ні у CBC-режимі, ні в інших режимах;

3) *цілісність IV у CBC.* Попри те, що вектор ініціалізації *IV* у CBC-режимі не секретний, має бути забезпечено його цілісність, оскільки навмисна модифікація *IV* дозволяє зловмиснику виконувати передбачувані зміни бітів у розшифрованому першому блоці відкритого тексту. Використання секретних *IV* є одним із методів запобігання таким модифікаціям. Однак, якщо потрібно забезпечити цілісність повідомлення, необхідно використовувати інші механізми, тому що механізми шифрування, переважно, гарантують тільки конфіденційність.

Зворотне завантаження шифротексту – CFB (Cipher Feedback). У режимі CFB також відбувають "маскування" блока відкритого тексту вже зашифрованими блоками:

$$c_i = E_k(m_i \oplus c_{i-1}),\ \ m_i = D_k(c_i) \oplus c_i\,. \qquad (2.3)$$

За своїми можливостями цей режим схожий на режим CBC. Проте якщо довжина повідомлення не кратна розміру блока шифру, то в режимі CBC необхідно доповнювати останній блок додатковими бітами та повідомляти на приймальну сторону справжній розмір повідомлення, а режим CFB дозволяє сформувати шифротекст того самого розміру, що й вихідне повідомлення (рис. 2.5).

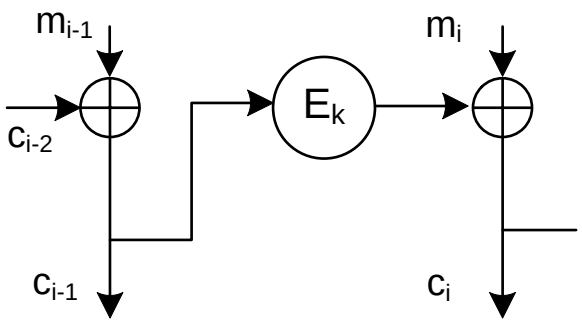


Рис. 2.5. **Схема режиму CFB**

У CFB-режимі на вхід подають k-бітовий ключ K, n-бітовий IV, послідовність r-бітових блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$, де $1 \le r \le n$. На виході формують r-бітові блоки шифротексту $C = c_1, c_2, \ldots, c_u \ldots$.

Спочатку в буфер зворотного зв'язку записують значення IV - $I_1 = IV$. Потім для всіх $1 \le i \le u$ виконують такі операції:

1) обчислення функції шифрування:

$$O_i = E_k(I_i); \tag{2.4}$$

2) вибору r лівих бітів результату шифрування O_i:

$$t_i = O_i \sim r, \tag{2.5}$$

де $O_i \sim r$ означає операцію вибірки;

3) формування та передання r-бітового блока шифротексту c_i:

$$c_i = p_i \oplus t_i; \tag{2.6}$$

4) формування змінного зворотного зв'язку із блока шифротексту:

$$I_{i+1} = 2^r I_i \oplus c_i \bmod 2^n. \tag{2.7}$$

Ця операція еквівалентна зсуву значення буферного регістру I_i на r-бітів праворуч і запису в кінець регістра r-бітового блока c_i. Розшифрування виконують відповідно:

$$P_i = c_i \oplus t_i; \;\; i = 1, \ldots, u, \tag{2.8}$$

де t_i, O_i і I_i обчислюють, відповідно до виразів, наведених раніше

$$I_1 = IV.$$

На рис. 2.6 подано схему CFB-режиму.

Слід зазначити, що необхідно мати ще й буфер зворотного зв'язку для зберігання c_{i-1}, що звичайно дорівнює n.

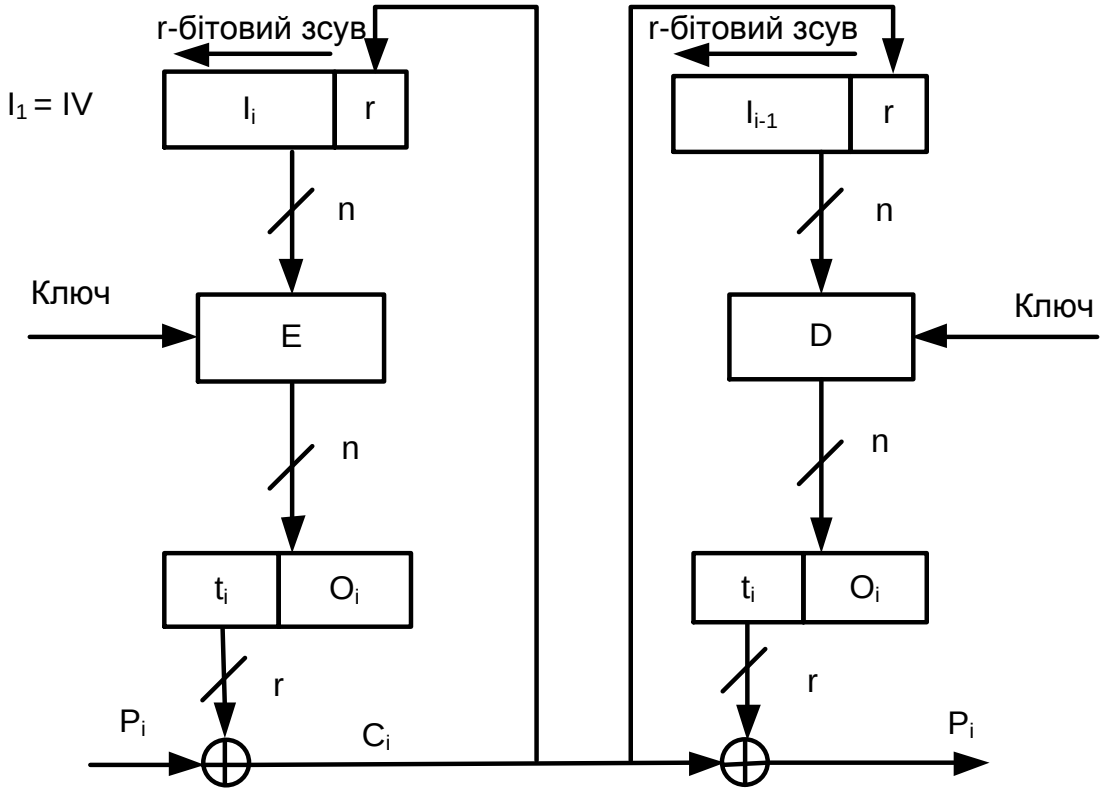


Рис. 2.6. **Режим зворотного зв'язку за шифротекстом**

CFB-режим має такі властивості:

1) як і у CBC-режимі шифрування, зміна *IV* приводить до різних результатів шифрування однакового відкритого тексту. Значення *IV* не обов'язково має бути секретним, однак це бажано в деяких застосунках;

2) аналогічно до CBC-режиму, механізм зчеплення призводить до того, що блок шифротексту c_i залежить від двох блоків відкритого тексту p_i і p_{i-1}. Отже, перевпорядкування блоків шифротексту впливає на дешифрування. Правильне дешифрування правильних блоків шифротексту потребує визначення [n/k] правильних попередніх блоків шифротексту для того, щоб регістр зворотного зв'язку містив правильне значення;

3) одна або більше помилок у будь-якому одиничному *r*-бітовому блоці шифротексту c_i впливає на дешифрування наступних [n/k] блоків шифротексту, тобто поки не буде закінчено оброблення *n* бітів шифротексту, після яких помилковий блок c_i буде витиснено з регістра зворотного зв'язку. Відновлений відкритий текст p_i' буде відрізнятися від p_i саме в тих позиціях, у яких відбулися помилки в c_i. Інші некоректно відновлені

блоки відкритого тексту, звичайно, будуть випадковими векторами, тобто будуть мати 50 % помилкових бітів. У такий спосіб зловмисник може здійснити передбачувані зміни бітів у p_i, зміниши відповідні біти у c_i;

4) CFB, як і CBC-режим, самосинхронізується, але потребує [n/k] блоків шифротексту для відновлення синхронізації;

5) для $r < n$ продуктивність режиму зменшують в n/r разів (порівняно із CBC), тому що кожне шифрування E забезпечує формування тільки r бітів шифротексту на виході.

Розглянутий режим є режимом роботи з r-бітовими вхідними символами та r-бітовим зворотним зв'язком. Цей режим схожий із режимом CFB, стандартизованим для DES (NBS FIPS Pub 81 і ANSI X3.106), і його позначають як CFB r-бітовий символ / r-бітовий зворотний зв'язок. Стандарт ISO/IEC 10118:1991 визначає більш загальний CFB-режим. Використанням цього режиму можна здійснювати оброблення j-бітових блоків відкритого тексту за r-бітового зворотного зв'язку, причому $j \le r$. Таким чином для реалізації режиму необхідно вибрати величину зворотного зв'язку r ($1 \le r \le n$) і величину блока відкритого тексту j ($1 \le j \le r$).

На вхід надходить k-бітовий ключ K, n-бітовий IV, послідовність j-бітових блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$. На виході алгоритму формують j-розрядні блоки шифротексту $C = c_1, c_2, \ldots, c_u$.

Шифрування здійснюють у такий спосіб.

Перед шифруванням значення вектора ініціалізації $I_1 = IV$ записують у регістр зворотного зв'язку.

Для всіх $1 \le i \le u$ виконують такі операції:

1) обчислення функції шифрування:

$$O_i = E_k(I_i); \tag{2.9}$$

2) вибору j лівих бітів із вектора O_i:

$$t_i = O_i \sim j; \tag{2.10}$$

3) формування блока шифротексту:

$$c_i = p_j + t_i; \tag{2.11}$$

4) формування змінного зворотного зв'язку із блока шифротексту c_i:

$$F_i = F^{(1)}(r - j) \| C_i, \tag{2.12}$$

де $F^{(1)}(r - j)$ – блок із $r - j$ одиниць;

$\|$ – операція конкатенції;

5) I_i зсув регістра на r бітів уліво і запис F_i у регістр:

$$I_{i+1} = 2^r I_i + F_i \bmod 2^n. \tag{2.13}$$

Слід зазначити, що для $i = u$ операції пункти 4 і 5 не виконують.

Розшифрування виконують аналогічно, за винятком того, що:

$$P_i = c_i \oplus t_i. \tag{2.14}$$

На рис. 2.7 подано схематичне зображення функціонування алгоритму шифрування у CFB-режимі за ISO/IEC 10118: 1991.

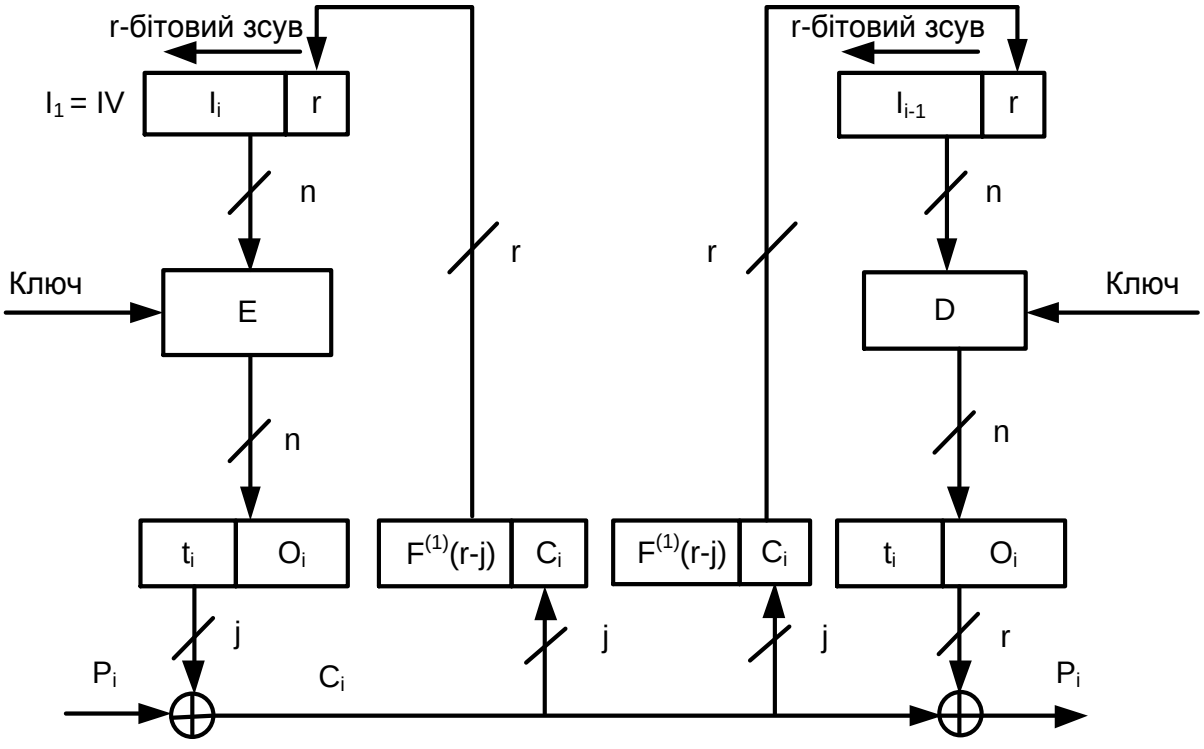


Рис. 2.7. **Режим CFB – *j*-бітовий відкритий текст / *r*-бітовий зворотний зв'язок (*j*/*r* CFB-режим)**

І, нарешті, у другій редакції стандарту ISO/IEC 10116: 1997 було запропоновано узагальнену версію CFB-режиму. Стандартизований режим забезпечує "конвеєрне оброблення даних".

У розглянутих версіях CFB-режиму результат шифрування одного блока відкритого тексту є входом для шифрування наступного блоку. Це означає, що неможливі "конвеєрні" обчислення, тобто неможливо розпочати шифрування одного блока до закінчення оброблення попереднього блока. Щоб уникнути цього недоліку, у нову версію CFB-режиму введено вже m-розрядний буфер зворотного зв'язку, де $2^n \geq m \geq n$. Також для роботи необхідно m-розрядний IV.

На вхід алгоритму надходять: k-розрядний ключ K, m-розрядний IV, послідовність j-розрядних блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$. На виході алгоритму формують послідовність j-розрядних блоків закритого тексту $C = c_1, c_2, \ldots, c_u$. Спочатку здійснюють запис у буфер зворотного зв'язку (регістр зсуву I_i). Значення вектора ініціалізації $I_1 = IV$.

Далі для всіх $i = \overline{1,u}$ виконують такі операції:

1) вибору n бітів із лівого боку регістра зсуву I:

$$X_i = I_i \sim n; \tag{2.15}$$

2) обчислення функції шифрування E:

$$O_i = E_k(X_i); \tag{2.16}$$

3) вибору j бітів із лівого боку O_i:

$$t_i = O_i \sim j; \tag{2.17}$$

4) формування та передавання блока зашифрованого тексту:

$$C_i = p_i \oplus t_i; \tag{2.18}$$

5) формування змінного зворотного зв'язку із блока шифротексту c_i доповненням його зліва $r - j$ одиницями:

$$F_i = F^{(1)}(r - j) | c_i; \tag{2.19}$$

6) зсув умісту буфера зворотного зв'язку *I* на *r* розрядів уліво й запис F_i у регістр:

$$I_{i+1} = 2^r I_i + F_i \bmod 2^n. \qquad (2.20)$$

Дешифрування виконують аналогічним чином, за винятком операції 4, а саме:

$$P_i = c_i \oplus t_i. \qquad (2.21)$$

На рис. 2.8 схематично подано CFB-режим за ISO/IEC 10116. Необхідно зазначити, що у другій редакції ISO/IEC 10116 рекомендовано вибирати $j = r$.

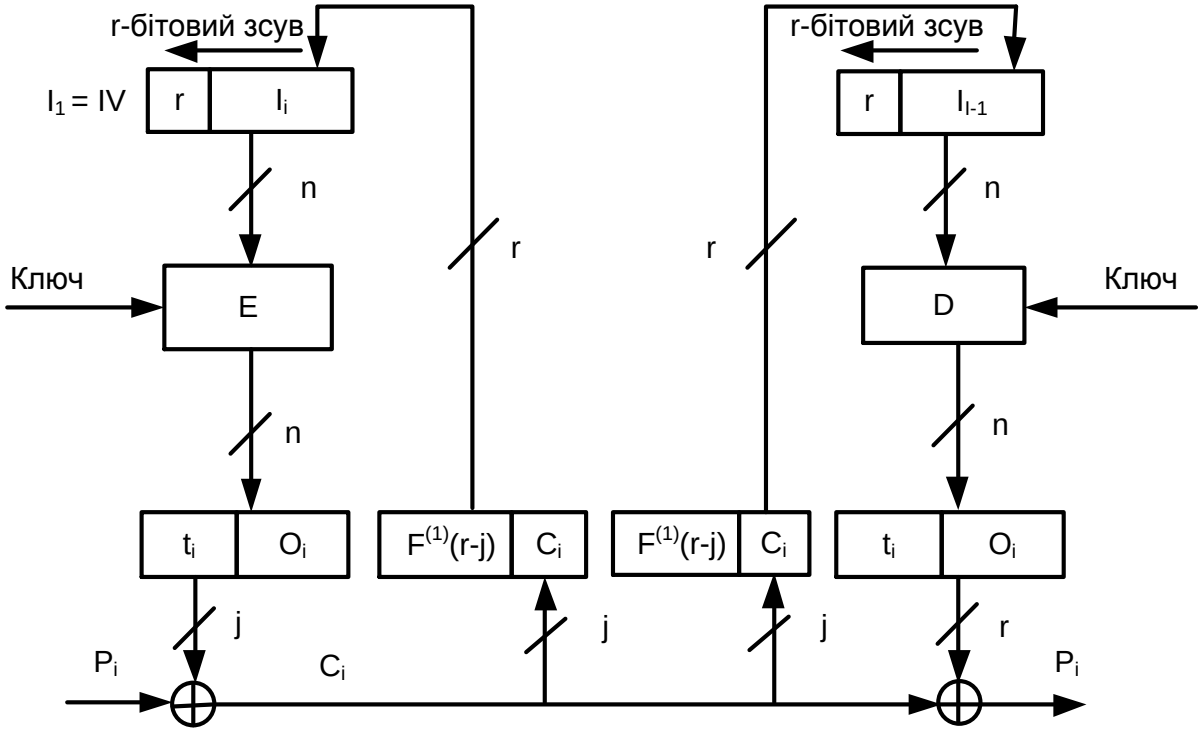


Рис. 2.8. **Режим CFB за ISO/IEC 10116 із конвеєрним обробленням даних**

Зворотне завантаження вихідних даних – OFB (Output Feedback). У режимі OFB вихідне повідомлення не піддають криптоперетворенню, його додають за модулем 2 із шифрованими на секретному ключі блоків s_i (s_0 є несекретним параметром режиму):

$$c_i = m_i \oplus s_i, \;\; m_i = c_i \oplus s_i, \;\; s_i = E_k(s_{i-1}). \qquad (2.22)$$

У цьому режимі, як і в режимі ECB, помилки, які можуть виникнути під час передавання шифротексту каналами зв'язку, локалізуються у блоці, не поширюючись на сусідні, причому в режимі OFB помилковими будуть тільки біти, які піддавали зміні (в ECB зміниться весь блок). Це дає можливість зловмисникові непомітно для приймальної сторони підмінити блок шифротексту. Можливості розпаралелювання процедур шифрування/дешифрування утруднено (рис. 2.9).

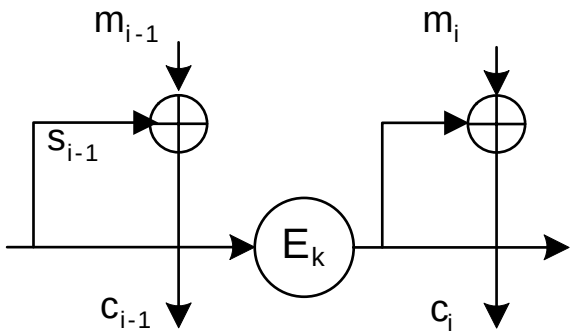


Рис. 2.9. **Можливості розпаралелювання процедур шифрування/дешифрування**

Цей режим подібний до CFB. Різниця полягає в тому, що вихід алгоритму в режимі OFB подають знову в регістр зсуву, тоді як у режимі CFB у регістр зсуву подають результат застосування операції XOR до незашифрованого блока та результату алгоритму.

Основна перевага режиму OFB полягає в тому, що якщо під час передавання відбулася помилка, то вона не поширюється на наступні зашифровані блоки. Тим самим зберігають можливість дешифрування наступних блоків. Наприклад, якщо з'являється помилковий біт у C_i, то це призведе тільки до неможливості дешифрування цього блока і визначення P_i. Подальшу послідовність блоків буде розшифровано коректно. За використання режиму CFB C_i подають як вхід у регістр і є причиною подальшого спотворення потоку.

Недолік OFB полягає в тому, що він більш уразливий до атак модифікування потоку повідомлень, ніж CFB.

Режим OFB може бути використано для застосувань, у яких має бути виключено будь-яке поширення помилок. Режим подібний до режиму CFB і дозволяє шифрувати блоки різної довжини, але як зворотний зв'язок

використовують не блок шифротексту, а шифрований блок із виходу функції *E*.

Поширено дві версії OFB-режиму роботи *n*-розрядного блокового шифру. Більш стійка версія ISO/IEC 10116: 1997 (ISO/IEC 10118: 1991) потребує *n*-розрядного зворотного зв'язку (повний зворотній зв'язок). Раніше було взято версію NBS FIPS 81, що працює з $r \le n$-розрядним зворотним зв'язком. На рис. 2.10 подано схему OFB-режиму за ISO/IEC 10116: 1997. У цій версії OFB-режиму на вхід алгоритму подають:

k-бітовий ключ *K*, *n*-розрядний *IV*, послідовність *r*-розрядних блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$, $1 \le r \le n$. На виході формують послідовність *r*-розрядних блоків шифротексту. Під час дешифрування відновлюють відкритий текст.

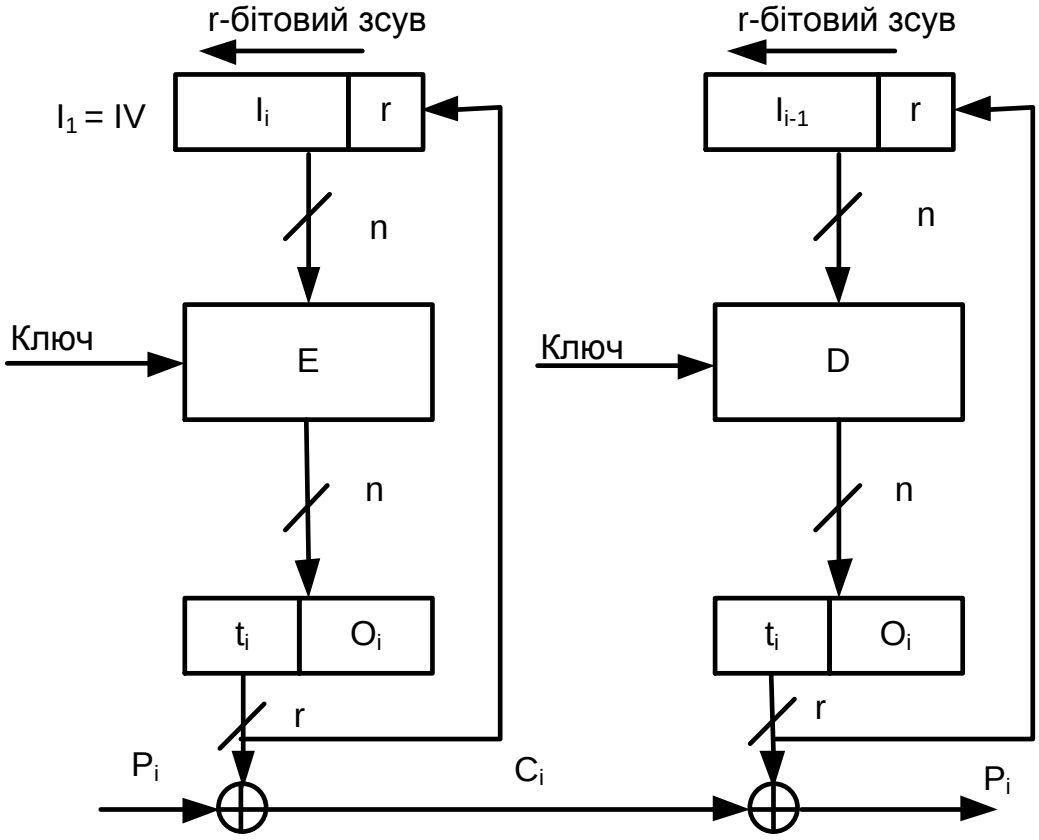


Рис. 2.10. **Режим OFB – *r*-бітовий відкритий текст / *n*-бітовий зворотний зв'язок (*r*/*n* OFB-режим)**

Шифрування здійснюють у такий спосіб.

Спочатку в регістр I_i записують значення *IV*. Потім для $1 \le i \le u$ виконують такі операції:

1) обчислення шифрованого блока:

$$O_i = E_k(I_i); \tag{2.23}$$

2) вибору з величини O_i r бітів зліва:

$$t_i = O_i \sim r; \tag{2.24}$$

3). передавання r-розрядного блока шифротексту c_i до каналу зв'язку:

$$c_i = p_i \oplus t_i; \tag{2.25}$$

4) відновлення вмісту регістра зсуву I_i:

$$I_{i+1} = O_i. \tag{2.26}$$

Під час розшифрування встановлюють $I_1 = IV$ і для всіх $1 \le i \le u$, здійснюють ті самі дії, за винятком операції 3:

$$p_i = c_i \oplus t_i. \tag{2.27}$$

Реалізацією режиму OFB за NBS FIPS B1 на вхід алгоритму подають k-бітовий ключ K, n-розрядний IV, послідовність r-розрядних блоків відкритого тексту $P = p_1, p_2, \ldots, p_u$, де $1 \le r \le n$. На виході формують послідовність r-розрядних блоків шифротексту $C = c_1, c_2, \ldots, c_u$.

Алгоритм шифрування та розшифрування аналогічний розглянутим раніше за винятком того, що операцію відновлення регістра $I_{i+1} = O_i$ замінюють на операцію зсуву регістра та запису в нього значення t_i:

$$I_{i+1} = 2^r I_i + F_i \bmod 2^n. \tag{2.28}$$

Необхідно розглянути властивості OFB-режиму:

1) як у CBC і CFB-режимах шифрування, зміна IV приводить до різних результатів шифрування однакового відкритого тексту;

2) ключовий потік не залежить від відкритого тексту;

3) одна або більше однобітових помилок у будь-якому символі шифротексту c_i впливає на дешифрування тільки цього символу точно в тих самих бітових позиціях, у яких виявили помилки в c_i. У такий спосіб забезпечено повне відновлення бітів відкритого тексту;

4) *OFB*-режим відновлюється після помилок у бітах шифротексту, але не може самосинхронізуватися після втрати бітів шифротексту, які руйнують вирівнювання (синхронізацію) розшифрувального ключового потоку. У цих випадках необхідна пересинхронізація шифру;

5) для $r < n$ продуктивність зменшується в n/r разів (як і у CFB-режимі). Однак в усіх випадках, оскільки ключовий потік не залежить від відкритого й шифрованого тексту, його може бути обчислено заздалегідь за заданим ключем і *IV*;

6) несекретний вектор ініціалізації має змінюватися, якщо в OFB-режимі повторно використовують ключ *K*. Інакше буде сформовано ідентичний ключовий потік, і шляхом додавання за модулем 2 (*XOR*) відповідних шифротекстів зловмисник може скористатися з того, що шифр буде з нескінченним ключем, де як ключ використовують відкритий текст.

У такий спосіб, виходячи із властивостей режиму, на практиці OFB-режим використовують тільки для забезпечення конфіденційності. У спрощеному OFB-режимі виводять відновлення вхідного блока як функцію лічильника, тобто: $I_{i+1} = I_{i+1}$.

Це дозволяє уникнути проблеми так званого короткого циклу та забезпечити відновлюваність після помилок в обчисленні *E*. Більш того, це забезпечує властивість випадкового доступу: не обов'язково дешифрувати *i*-й блок шифротексту для того, щоб дешифрувати *i* + 1-й блок.

Раніше версія ISO-режиму була більш стійкою, ніж версія NBS FIPS. В OFB-режимі з повним *n*-розрядним зворотним зв'язком ключовий потік генерують із використанням ітеративної функції $O_i = E_k (O_i - 1)$. E_k є перестановкою, тому у припущенні, що *k* є випадковою величиною, E_k дійсно є випадковим вибором із множини $(2^n)!$ перестановок за *n* елементами. Може статися, що для фіксованих (випадкових) значень ключа та вектора ініціалізації очікувана довжина циклу перед повторенням будь-якого значення O_i буде дорівнювати $2^n - 1$. З іншого боку, якщо кількість бітів зворотного зв'язку дорівнює $r < n$, як визначено у FIPS 81, то ключовий потік буде формуватися з використанням ітерації $O_i = f (O_i - 1)$, де *f* не є функцією перестановки. У припущенні, що вона поводить себе як випадкова функція, очікувана довжина циклу становить величину порядку $2^{n/2}$. Отже,

кращим є використання OFB-режиму з повним *n*-розрядним зворотним зв'язком.

Необхідно зазначити, що й OFB-режим із повним зворотним зв'язком, і режим лічильника забезпечують можливість застосування блокового шифру як генератора ключового потоку для потокового шифру. Аналогічним чином і у CFB-режимі здійснюють шифрування потоку символів, а блоковий шифр використовують як генератор, залежний від відкритого тексту.

У табл. 2.1 відображено ефект поширення помилки в режимах шифрування.

Таблиця 2.1

Ефекти поширення помилки

Режими шифрування	Ефекти від помилки в біті шифротексту c_i	Ефекти від помилки в *i*-му біті вектора ініціалізації *IV*
ECB	ПЛБ[1] за розшифрування блока c_i	Не використовують
CBC	ПЛБ[1] за розшифрування блока c_i ПОБ[2] за розшифрування блока c_{i+1}	ПОБ[2] за розшифрування блока c_1
CFB	ПОБ[2] за розшифрування блока c_i ПЛБ[1] за розшифрування блоків c_{i+1}, ..., $c_{i+n/s}$	ПЛБ[1] за розшифрування блоків, $c_1, c_2, ..., c_i$ де $1 \le i \le n - i / s$
OFB	ПОБ[2] за розшифрування блока c_i	ПЛБ[1] за розшифрування блоків $c_1, c_2, ..., c_t$

Примітки:

1 – ПЛБ – помилка в будь-якому біті, тобто ушкодженим за розшифрування з імовірністю 1/2 може виявитися будь-який біт блока (або порції);

2 – ПОБ – помилка в одному біті, тобто за розшифрування ушкоджують один біт, що перебуває в тій самій позиції, що й біт, який спричинив помилку.

Алгоритм AES

Основні параметри:

- симетричний блоковий шифр зі змінним розміром вхідного блока: 128, 192 і 256 бітів;
- довжина ключа також є змінною, набуває тих самих значень;

• кількість циклів оброблення становить 10, 12, 14, що залежить від довжини блока та ключа.

• AES за один етап обробляє увесь вхідний блок, а не його половину, як це виконують у фейстелівських алгоритмах.

Алгоритм *AES* складається з послідовності операцій, що виконують над проміжним масивом 4 × 4 байтів, який називають *станом* і позначають $S = (S_{i,j})$. На початку процесу шифрування 16 байтів стану (128 бітів) ініціалізують байтами відкритого тексту p_i:

$$\begin{matrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{matrix} \xrightarrow{S = P} \begin{matrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{matrix}$$

Алгоритм ініціалізації, як і більшість наступних операцій, оптимізовано для програмної реалізації на 32-бітових платформах, оскільки 4 байти кожного стовпчика можуть обробляти як одне 32-бітове слово.

Після ініціалізації та додавання циклових ключів стан перетворюють за допомогою N_r послідовних операцій раундової функції з невеликою модифікацією останнього циклу. Повний алгоритм шифрування можна описати таким псевдокодом:

```
S=AddRoundKey (P,W0)
for i=1 toNr – 1 do
S=SubBytes (S)
S=ShiftRows (S)
S=MixColumns (S)
S=AddRoundKey (S,Wi)
endfor
S=SubBytes (S)
S=ShiftRows (S)
C=AddRoundKey (S,WNr)
```

Кількість циклів N_r залежить від довжини ключа та має фіксовані значення 10, 12, 14 для *AES-128, AES-192 і AES-256*, відповідно. Також зазначте, що перетворення *MixColumns* в останньому циклі відсутнє.

Алгоритм DES

Основні параметри:

- симетричний блоковий шифр, розмір блока – 64 біти;
- довжина ключа – 56 бітів (звичайно використовують 64-бітове число, але кожний восьмий біт використовують для перевірки парності й ігнорують);
- є 16-раундовою збалансованою мережею Фейстеля із двома додатковими перестановками.

Структурну схему циклової функції алгоритму DES подано на рис. 2.11, а схему повного алгоритму DES – на рис. 2.13.

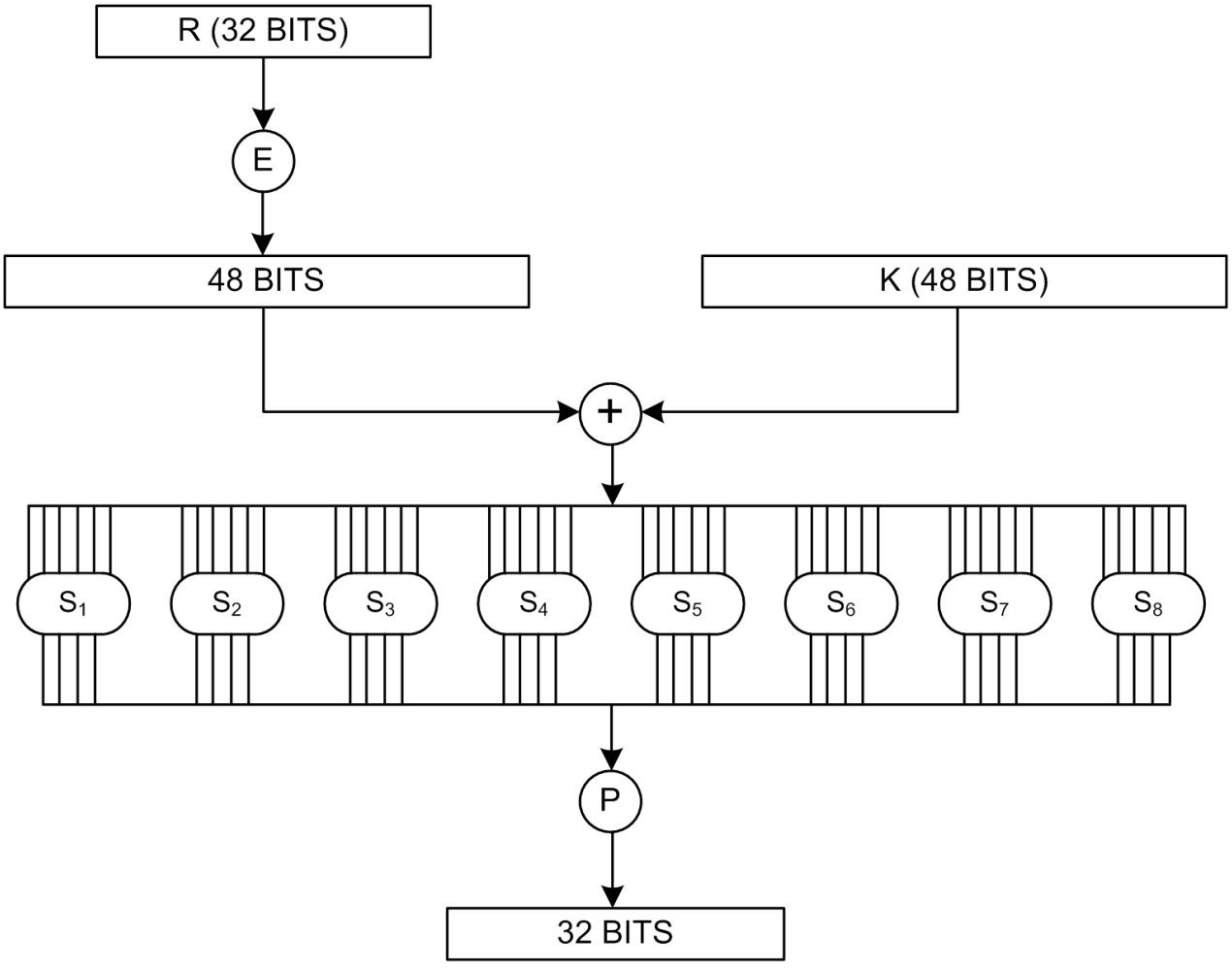


Рис. 2.11. **Структурна схема циклової функції**

Нелінійну заміну задають таблицями замін. Наприклад, для першого S-блока таблиця замін має вигляд, поданий на рис. 2.12:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Рис. 2.12. **Таблиця замін**

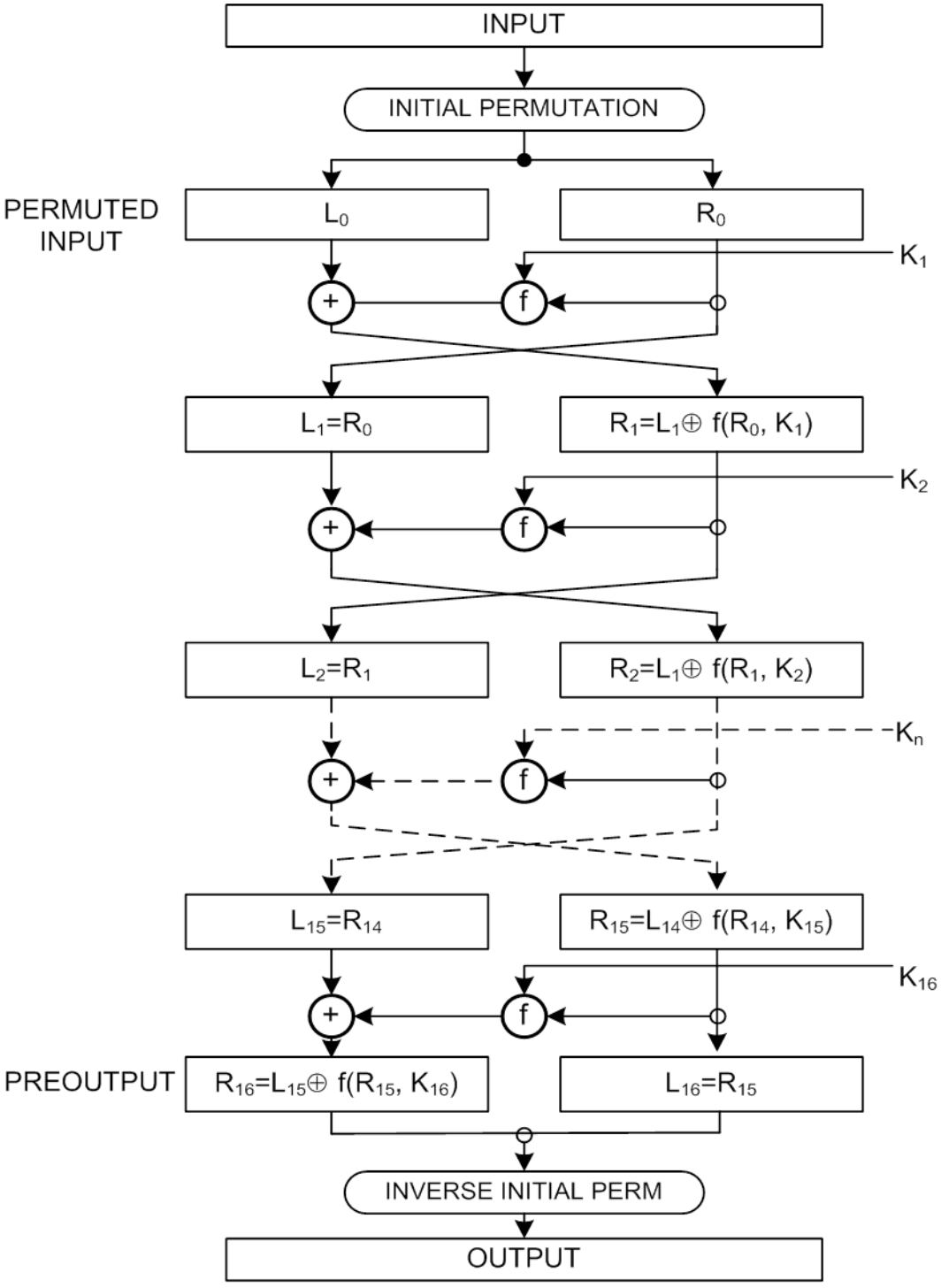


Рис. 2.13. **Структурна схема алгоритму DES Triple DES**

Є багато різних варіантів потрійного DES. Найбільш популярними з них є два: *3DESEDE2* (Encrypt-Decrypt-Encrypt із двома ключами) та *3DESEDE3* (Encrypt-Decrypt-Encrypt із трьома ключами).

Потрійний DES із двома ключами. У цьому алгоритмі використовують два ключі по 56 бітів, тобто загальна довжина ключа дорівнює 112 бітам. Шифрування цим алгоритмом передбачає етапи, показані на рис. 2.14.

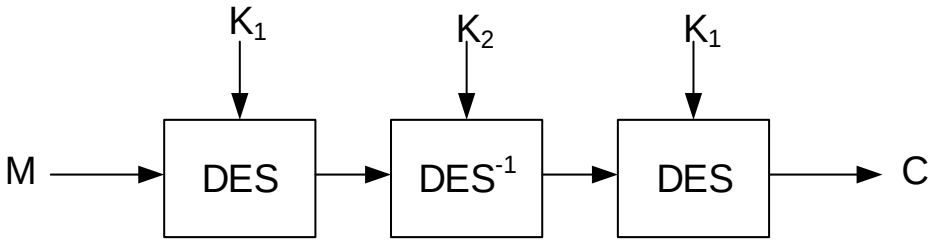


Рис. 2.14. **Схема шифрування алгоритму *3DESEDE2***

Як показано на рис. 2.14, відкрите повідомлення *M* спочатку шифрують звичайним однократним DES на ключі K_1, потім розшифровують на ключі K_2, після чого знов шифрують – на K_1. У цьому разі зростання криптостійкості досягають як збільшенням довжини загального ключа (до 112 бітів + біти парності), так і кількістю циклів оброблення. На відміну від однократного, потрійний DES еквівалентний 48 раундам оброблення відкритого тексту. Очевидно, що потрійний DES саме втричі повільніший за звичайний, хоча й не такий повільний, як асиметричні алгоритми. Однак швидкий розвиток комп'ютерної техніки дещо згладжує цей недолік. Етап розшифрування на ключі K_2 подано для сумісності з однократним DES у разі $K_1 = K_2$.

Розшифрування відбувається оберненим чином: на вхід алгоритму подають зашифрований текст (*C*); на першому етапі розшифровують на ключі K_1; на другому – шифрують на K_2; на третьому – знов розшифровують на K_1. У результаті визначають розшифровану інформацію (*M*).

Потрійний DES із трьома ключами. Відмінність від попереднього алгоритму полягає в тому, що тут використовують три ключі шифрування, отже, стійкість системи до атаки "грубою силою" зростає. Загальна довжина ключа досягає 56 × 3 = 168 бітів + біти парності. У разі $K_1 = K_2 = K_3$ *3DESEDE3* перетворюється на однократний DES, правда, утричі повільніший.

Схему шифрування цим алгоритмом показано на рис. 2.15.

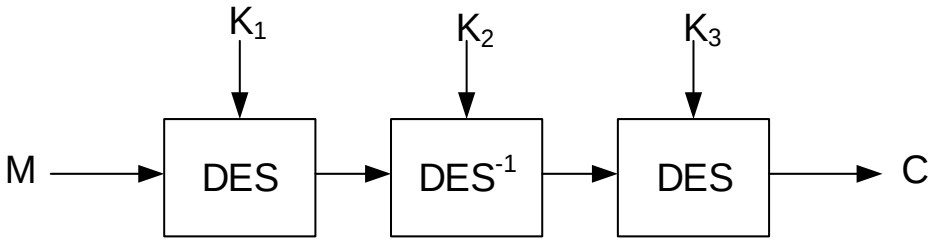


Рис. 2.15. **Потрійний DES із трьома ключами**

Розшифрування виконують аналогічно: спочатку шифроване повідомлення розшифровують на ключі K_3, потім зашифровують на ключі K_2, і, нарешті, знов розшифровують, але на ключі K_1.

Падіння швидкодії під час роботи 3DES іноді дуже помітне, і, наприклад, у режимі зчеплення блоків це сповільнення не вдається компенсувати додатковим апаратним обладнанням. У багатьох випадках, наприклад, під час шифрування критичних каналів зв'язку, таке зменшення продуктивності неприпустиме.

Алгоритм Blowfish

Алгоритм Blowfish є 16-раундовою мережею Фейстеля з довжиною блока 64 біти. Ключ може мати довільну довжину в межах 448 бітів. Хоча перед початком шифрування виконують складну фазу ініціалізації, саме шифрування даних виконують досить швидко. Алгоритм призначено в основному для операцій, де ключ міняють нечасто, до того ж є фаза початкової автентифікації сторін, під час якої відбувається погодження загальних параметрів шифрування. Реалізація на 32-бітових мікропроцесорах із великим кешем даних Blowfish значно швидша за DES.

Алгоритм складається із двох частин: розгортання ключа та шифрування даних. Розгортання ключа перетворює ключ довжиною, принаймні 448 бітів, на кілька масивів підключів загальною довжиною 4 168 байтів. Кожний раунд шифру складається з перестановки, залежної від ключа, і заміни, яка залежить і від ключа, і від даних. Раундовими операціями є *XOR* і додавання 32-бітових слів.

Blowfish використовує велику кількість підключів, які необхідно обчислити заздалегідь, до початку процесів шифрування або розшифрування даних. Алгоритм містить:

1) P – масив, що містить 18 32-бітових підключів:

$$P_1, P_2, ..., P_{18};$$

2) чотири 32-бітових S-бокси із 256 входами кожний. Перший індекс означає номер S-боксу, другий індекс – номер входу:

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$
$$S_{2,0}, S_{2,1}, \ldots S_{2,255};$$
$$S_{3,0}, S_{3,1}, \ldots S_{3,255};$$
$$S_{4,0}, S_{4,1}, \ldots S_{4,255}.$$

Метод, який використовують для обчислення цих підключів, описано.

Процес шифрування

Входом алгоритму є 64-бітовий блок даних *X*, який розподіляють на дві 32-бітові половини: *XL* і *XR*.

XL = XL XOR Pi.

XR = F (*XL*) *XOR XR.*

Swap XL and XR.

Структура функції F

Розподілити *XL* на чотири 8-бітових елементи *A, B, C, D*:

$$F(XL) = ((S1,A + S2,B \bmod 232) \; XOR \; S3,C) + S4,D \bmod 232.$$

Розшифрування відрізняється від шифрування тим, що *Pi* використовують в оберненому порядку.

Генерування підключів

Підключі обчислюють із використанням самого алгоритму *Blowfish* у такий спосіб:

1) ініціалізують перший *P*-масив і чотири *S*-бокси фіксованим рядком;

2) виконують операцію *XOR* P_1 з першими 32 бітами ключа, операція *XOR* P_2 із другими 32 бітами ключа і т. д. Цикл повторюють доти, поки увесь *P*-масив не буде побітово підсумовано з усіма бітами ключа. Для коротких ключів виконують конкатенацію ключа із самим собою;

3) шифрують нульовий рядок алгоритмом Blowfish, використовуючи підключі, описані в пунктах 1 і 2;

4) P_1 і P_2 замінюють на результат, досягнутий на кроці 3;

5) вихід кроку 3 шифрують з використанням алгоритму Blowfish із модифікованими підключами;

6) P_3 і P_4 замінюють на результат, досягнутий на кроці 5;

7) процес продовжують заміною всіх елементів *P*-масиву, а потім усі чотири *S*-бокси – результатами шифрування відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібно виконати 521 ітерацію.

Алгоритм CAST-256

Алгоритм шифрування CAST-256 має DES-подібну SPN-структуру, побудовану на базі алгоритму CAST-128, що має високу стійкість до диференціального, лінійного криптоаналізу та криптоаналізу на зв'язаних ключах. Цей шифр також має низку інших позитивних криптографічних властивостей, включаючи точний лавинний критерій, критерій незалежності бітів (bit independence criterion – BIC), властивість недоповнюваності та відсутність слабких і напівслабких ключів. Він є гарним кандидатом для багатоцільового використання в інтернеті, де потрібен криптографічно сильний та доступний алгоритм.

CAST-256 має розмір блока 128 бітів і різні довжини ключів (128, 160, 192, 224 або 256 бітів).

Структура алгоритму. Такі описи CAST-128 справедливі й для CAST-256. CAST-128 використовує пару підключів на раунд: 5-бітову змінну Kr_i використовують як "ротаційний" ключ для *i*-го раунду та 32-бітову змінну Km_i використовують як "маскувальний" ключ для раунду *i*.

CAST-128 використовує три різні раундові функції, як це описано в табл. 2.2.

Зазначте, що "+" і "-" – це додавання та віднімання за модулем 2^{32}, "^" – побітове додавання за модулем *2*, *i* <<< – циклічна операція зсуву вліво.

Таблиця 2.2

Раундові функції CAST-128

Тип 1	$I = ((Km_i + D) <<< Kr_i)$ $O = ((S_1I_a \wedge S_2I_b) - S_3Ic) + S_4I_d$
Тип 2	$I = ((Km_i \wedge D) <<< Kr_i)$ $O = ((S_1I_a - S_2I_b) + S_3I_c) \wedge S_4I_d$
Тип 3	$I = ((Km_i - D) <<< Kr_i)$ $O = ((S_1I_a + S_2I_b) \wedge S_3I_c) - S_4I_d$

Примітки:

D – вхід цієї операції; *Ia* – найбільший байт *I*; *Id* – найменший байт *I*; S_i – це *i*-й *S*-блок, *O* – це вихід операції.

CAST-128 використовує чотири S-блоки раундових функцій – S_1 – S_4.

Опис CAST-256. Нехай раундові функції *f1, f2, f3* буде визначено так само, як і в CAST-128.

Припустіть:

BETA = (*ABCD*) буде 12-бітовим блоком, де *A*, *B*, *C* і *D* мають довжину 32 біти.

"*BETA* <- *Qi*(*BETA*)" буде коротким позначенням:

$$C = C \text{ ^ } f1(D, Kr_0\_(i), Km_0\_(i));$$
$$B = B \text{ ^ } f2(C, Kr_1\_(i), Km_1\_(i));$$
$$A = A \text{ ^ } f3(B, Kr_2\_(i), Km_2\_(i));$$
$$D = D \text{ ^ } f1(A, Kr_3\_(i), Km_3\_(i)).$$

"*BETA* <- *QBARi*(*BETA*)" позначає:

$$D = D \text{ ^ } f1(A, Kr_3\_(i), Km_3\_(i));$$
$$A = A \text{ ^ } f3(B, Kr_2\_(i), Km_2\_(i));$$
$$B = B \text{ ^ } f2(C, Kr_1\_(i), Km_1\_(i));$$
$$C = C \text{ ^ } f1(D, Kr_0\_(i), Km_0\_(i));$$

(*Q*(*) – "пряма четвірка-раунд" і *QBAR*(*) – "реверсна четвірка-раунд").

*Kr_*(*i*) = {*Kr0_*(*i*), *Kr1_*(*i*), *Kr2_*(*i*), *Kr3_*(*i*)} – множина ротаційних ключів для *i*-ї четвірки-раунду, де $Kr_j\_(i)$ – це 5-бітовий ротаційний ключ для *f1, f2* або *f3* (як визначено в табл. 2.2);

*Km_*(*i*) = {$Km_0\_(i)$, $Km_1\_(i)$, $Km_2\_(i)$, $Km_3\_(i)$} – множина маскувальних ключів для *i*-ї четвірки-раунду, де *Kmj_*(*i*) – це 32-бітовий маскувальний ключ для *f1, f2* або *f3* (як визначено раніше);

KAPPA = (*ABCDEFGH*) – 256-бітовий блок, де *A*, *B*, ..., *H* мають розмір 32 біти.

"*KAPPA* <- *Wi*(*KAPPA*)" позначає таке:

$$G = G \text{ ^ } f1(H, Tr0\_(i), Tm0\_(i));$$
$$F = F \text{ ^ } f2(G, Tr1\_(i), Tm1\_(i));$$
$$E = E \text{ ^ } f3(F, Tr2\_(i), Tm2\_(i));$$
$$D = D \text{ ^ } f1(E, Tr3\_(i), Tm3\_(i));$$
$$C = C \text{ ^ } f2(D, Tr4\_(i), Tm4\_(i));$$
$$B = B \text{ ^ } f3(C, Tr5\_(i), Tm5\_(i));$$
$$A = A \text{ ^ } f1(B, Tr6\_(i), Tm6\_(i));$$
$$H = H \text{ ^ } f2(A, Tr7\_(i), Tm7\_(i));$$

(*W*(*) називають "прямим октетом";

"*Kr_*(*i*) <- *KAPPA*" позначає таке:

*Kr0_*(*i*) = *5LSB*(*A*), *Kr1_*(*i*) = *5LSB*(*C*), *Kr2_*(*i*) = *5LSB*(*E*), *Kr3_*(*i*) = *5LSB*(*G*), де *5LSB*(*x*) позначає п'ять найменш значущих бітів *x*.

"*Km_*(*i*) <- *KAPPA*" позначає:

*Km0_*(*i*) = *H*, *Km1_*(*i*) = *F*, *Km2_*(*i*) = *D*, *Km3_*(*i*) = *B*.

Шифрування CAST-256:

```
BETA = 128 бітів відкритого тексту:
for (i=0; i<6; i++)
BETA <- Qi(BETA)
        for (i=6; i<12; i++)
        BETA <- QBARi(BETA)
128 бітів шифротексту = BETA.
```

Зміна черговості раундових ключів для розшифрування. Шифр використовує 256-бітовий первинний ключ *K*. Розшифрування ідентичне шифруванню, за винятком того, що масиви ключів четвірок-раундів *Kr_*(*i*)*, Km_*(*i*)*,* визначених із ключа *K*, використовують у протилежному порядку:

```
for (i=0; i<12; i++)
{       KrNEW_(i) = Kr_(11-i)
        KmNEW_(i) = Km_(11-i)
}
```

Процедура розширення ключа CAST-256

```
Ініціалізація:
Cm = 2**30 * SQRT(2) = 5A827999 (base 16)
Mm = 2**30 * SQRT(3) = 6ED9EBA1 (base 16)
Cr = 19
Mr = 17
for (i=0; i<24; i++)
{ for (j=0; j<8; j++)
  { Tmj_(i) = Cm
    Cm = (Cm + Mm) mod 2**32
    Trj_(i) = Cr
    Cr = (Cr + Mr) mod 32
  }
}
```

```
Процедура розширення ключа:
KAPPA = ABCDEFGH = 256 бітів первинного ключа K.
for (i=0; i<12; i++)
 { KAPPA <- W2i(KAPPA)
   KAPPA <- W2i+1(KAPPA)
   Kr_(i) <- KAPPA
   Km_(i) <- KAPPA
 }
```

Зазначте:

($|K| = 128$) => ($E = F = G = H = 0$)

($|K| = 160$) => ($F = G = H = 0$)

($|K| = 192$) => ($G = H = 0$)

($|K| = 224$) => ($H = 0$)

Назва шифру. Для уникнення плутанини, коли використовують операцію зі змінним розміром ключа, назву CAST-256 розглядають як синонім CAST6; це дозволяє визначити розмір ключа без неоднозначності. Наприклад, CAST-256 зі 192-бітовим ключем позначають як CAST 6-192; коли ключ дорівнює 256 бітам, необхідно використовувати назву CAST 6-256.

Алгоритм IDEA

Перший варіант алгоритму *IDEA* з'явився 1990 року. Розробники алгоритму, Сюецзя Лай (Xuejia Lai) і Джеймс Мессі (James Massey), зі Швейцарського інституту ETH Zurich, дали йому назву PES (Proposed Encryption Standard – стандарт шифрування, що пропонують), оскільки цей алгоритм було запропоновано на заміну стандарту DES. Варто зазначити, що інститут ETH Zurich і професор Джеймс Мессі відомі також завдяки розробленим алгоритмам сімей *SAFER*, *SAFER+*, *SAFER++*.

За рік алгоритм модифікували для посилення криптостійкості до диференціального криптоаналізу. Нова версія дістала назву IPES (Improved PES – поліпшений PES), а ще через рік алгоритм змінив назву на IDEA (International Data Encryption Algorithm – міжнародний алгоритм шифрування даних).

Структура алгоритму. Алгоритм IDEA шифрує дані блоками по 64 біти, а ключ шифрування алгоритму має розмір 128 бітів. Блок шифрованих даних розподіляють на чотири 16-бітових підблоки *A*, *B*, *C* і *D*, над якими виконують вісім раундів перетворень:

$$A = A \times Kr_1;\ B = B + Kr_2;\ C = C + Kr_3;\ D = D \times Kr_4;\ T_1 = A + C;$$
$$T_2 = B + D;\ T_1 = T_1 \times Kr_5;\ T_2 = T_1 + T_2;\ T_2 = T_2 \times Kr_6;\ T_1 = T_1 + T_2;$$
$$A = A + T_2;\ B = B + T_1;\ C = C + T_2;\ D = D + T_1,$$

де Kr_n – підключ n раунду r ;

"+" – побітова логічна операція (XOR);

× – множення 16-бітових операндів за модулем $2^{16} + 1$, причому для значення нульового підблока також беруть значення за модулем 2 у 16-му степені.

Структуру алгоритму подано на рис. 2.16.

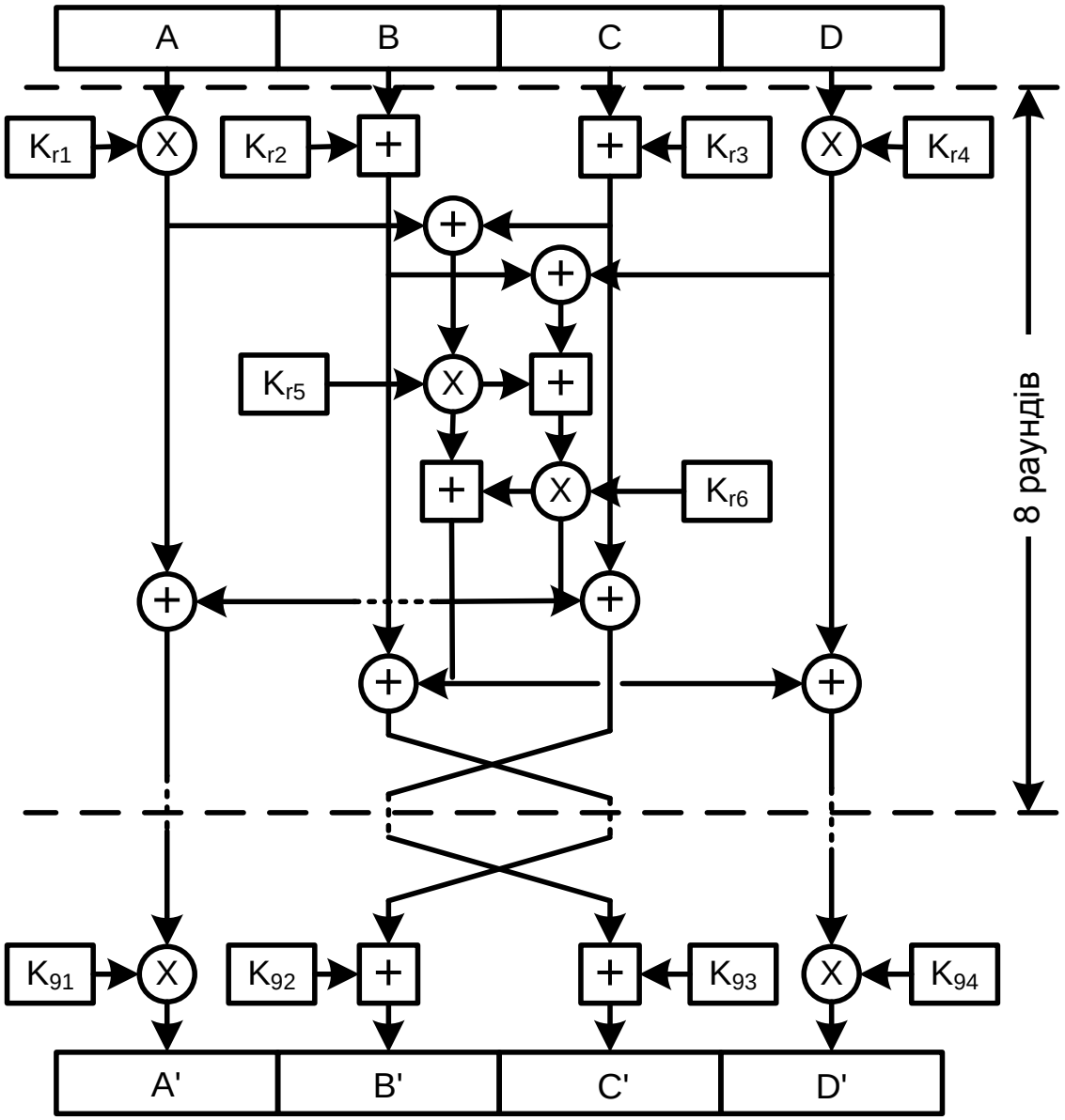


Рис. 2.16. **Структура алгоритму IDEA**

Після виконання описаних дій два внутрішні підблоки (*B* і *C*) міняють місцями – у всіх раундах, крім останнього. Після завершення восьми раундів виконують додаткові перетворення (іноді їх називають дев'ятим раундом алгоритму):

$$A' = A \times K9_1;\ B' = B + K9_2;\ C' = C + K9_3;\ D' = D \times K9_4.$$

Шифротекст є результатом конкатенації визначених значень *A'*, *B'*, *C'* і *D'*.

Операція розшифрування аналогічна до зашифрування з тією різницею, що в розшифруванні використовують модифіковані підключі в іншій послідовності:

$$K'r_1 = (K(10 - r)_1)^{-1};\ K'r_2 = -K(10 - r)_3;\ K'r_3 = -K(10 - r)_2;$$

$$K'r_4 = (K(10 - r)_4)^{-1};\ K'r_5 = K(9 - r)_5;\ K'r_6 = K(9 - r)_6,$$

за винятком раундів 1 і 9, у яких підключі $K'r_2$ і $K'r_3$ міняють місцями.

Тут $K'r_n$ – підключ n раунду розшифрування; r, x і x^{-1} – обернені значення x щодо описаних операцій додавання за модулем 2^{16} і множення за модулем $(2^{16} + 1)$, відповідно. До того ж $0\ \{y^{-1}\} = 0$.

Розгортання ключа. Завдання процедури розширення ключа – формування 52 16-бітових підключів, що використовують у раундах шифрування, та додаткових перетворень (тобто всього 832 біти ключової інформації). Ця процедура є досить простою, її виконують у такий спосіб:

- 128-бітовий ключ шифрування розподіляють на вісім підключів по 16 бітів; вони стають першими вісьмома підключами алгоритму ($K1_1$, $K1_2$, $K1_3$, $K1_4$, $K1_5$, $K1_6$, $K2_1$, $K2_2$);
- ключ шифрування циклічно зсувають уліво на 25 бітів;
- результат розподіляють на вісім наступних підключів;
- ключ шифрування циклічно зсувають уліво на 25 бітів і т. д. – до створення необхідної кількості підключів.

Криптостійкість алгоритму. Уже в наступному році після появи алгоритму PES його автори опублікували роботу, у якій було доведено слабкість алгоритму щодо диференціального криптоаналізу: для визначення ключа шифрування досить виконання 2^{64} операцій шифрування, тоді як атака "грубою силою" на 128-бітовий ключ потребувала б виконання 2^{128} операцій.

Алгоритм IDEA виник у результаті досить незначних модифікацій алгоритму PES (рис. 2.17). Схеми алгоритмів IDEA і PES відрізняються незначно:

операцію множення підблока *B* з іншим підключем раунду замінено операцією додавання;

операцію додавання підблока *D* із четвертим підключем раунду замінено операцією множення;

інакше виконують зсув підблоків наприкінці раунду.

Як зазначив один із найбільш відомих у світі фахівців-криптологів Брюс Шнайєр (Bruce Schneier) у праці "Прикладна криптографія", "… дивно, як такі незначні зміни можуть призвести до настільки великих розбіжностей". Алгоритм IDEA виявився фактично невразливим до диференціального криптоаналізу, у виданні 1996 р. Брюс Шнайєр відгукнувся про нього так: "Мені здається, це найкращий та найнадійніший блоковий алгоритм, опублікований дотепер".

Однак того самого 1996 року відомий криптоаналітик Пол Кохер (Paul Kocher) віднайшов досить складну атаку, що дозволяє шляхом багаторазових високоточних вимірювань часу виконання шифрування 2^{20} випадково вибраних відкритих текстів на ключах, зв'язаних певним співвідношенням із потрібним ключем, і наступного аналізу результатів, знайти ключ шифрування. Ця атака (як і інші атаки на зв'язаних ключах) передбачає, що криптоаналітик не має прямого доступу до ключа шифрування (наприклад, ключ прошитий у якому-небудь апаратному шифраторі або смарт-карті), але може змінювати певним чином різні його фрагменти.

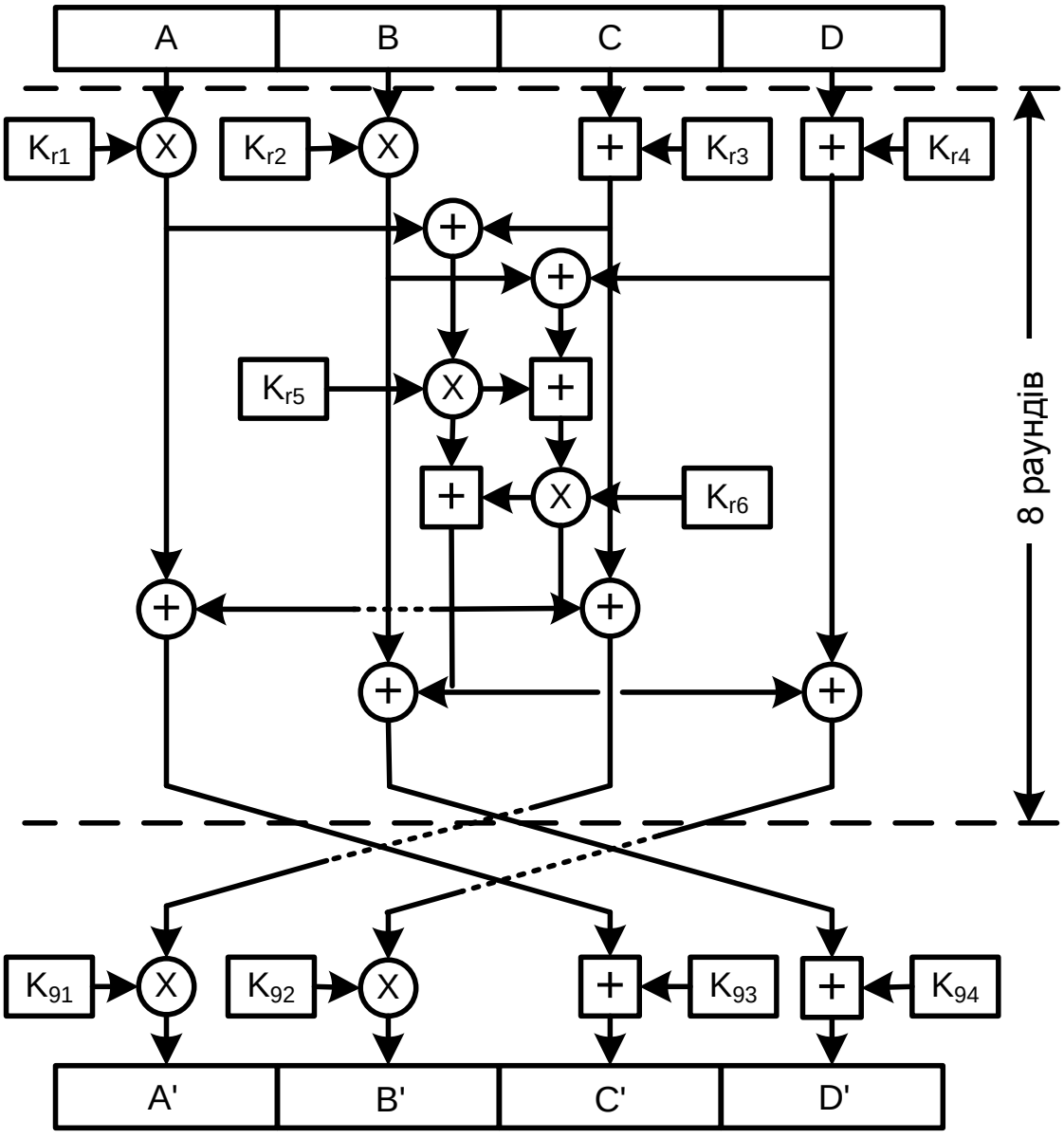


Рис. 2.17. **Структура алгоритму PES**

Раніше, 1993 року, декілька криптологів із Бельгії виявили в алгоритмі IDEA кілька класів слабких ключів (усього $2^{23} + 2^{35} + 2^{51}$ ключів різного ступеня слабкості), частину з яких, наприклад, можна обчислити криптоаналітичною атакою з підібраним відкритим текстом. Однак самі автори цієї атаки запропонували "протиотруту" – слабкі ключі вилучають накладанням операцією *XOR* спеціальної шістнадцяткової константи IDEA на кожний підключ перед його використанням.

Таким чином, незважаючи на виявлені недоліки, алгоритм IDEA вважають алгоритмом із високою криптостійкістю. Незаперечною ж перевагою цього алгоритму є висока швидкість зашифрування, як мінімум удвічі більша, ніж в алгоритмі DES (залежно від платформи, на якій виконують шифрування). А можливість виконання операції розширення ключа "на льоту" (тобто паралельно з виконанням раундів шифрування) дуже популярна й у сучасніших алгоритмах шифрування. Однак варто зауважити, що швидкість розшифрування трохи знижується через наявність ресурсомістких операцій обчислення мультиплікативних обернених величин за модулем ($2^{16} + 1$).

Алгоритм IDEA не став міжнародним стандартом шифрування, як того бажали його автори. Однак його можна вважати одним із найпоширеніших у світі алгоритмів шифрування. IDEA використовують дотепер у безлічі різних застосунків, зокрема в популярній програмі захисту даних PGP.

Алгоритм ГОСТ 28147-89

Стандарт шифрування даних Радянського Союзу ГОСТ 28147-89 було взято на озброєння 1989 року. В основі цього стандарту лежить мережа Фейстеля. Параметри цього алгоритму такі: довжина блока – 64 біти; довжина ключа – 256 бітів; кількість раундів – 32.

Процес шифрування:

Крок 0. *Визначення вихідних даних* для основного кроку криптоперетворення:

- N – перетворений 64-бітовий блок даних; у ході виконання кроку його молодша (N_1) і старша (N_2) частини обробляють як окремі 32-бітові цілі числа без знака. Таким чином, можна записати $N = (N_1, N_2)$;
- X – 32-бітовий елемент ключа;

Крок 1. *Додавання із ключем.* Молодшу половину перетвореного блока додають за модулем 2^{32} з елементом ключа, що використовують у цьому кроці.

Крок 2. *Поблокова заміна.* 32-бітове значення, визначене на попередньому кроці, інтерпретують як масив із восьми 4-бітових блокових кодів: $S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$. Значення кожного з восьми блоків заміняють на нове, яке вибирають за таблицею замін у такий спосіб: значення блока S_i замінюють на S_i-й елемент (нумерація з нуля) i-го вузла замін, тобто i-го рядка таблиці замін (нумерація також із нуля). Інакше кажучи, як заміну для значення блока вибирають елемент із таблиці з номером рядка, що дорівнює номеру блока, і номером стовпчика, що дорівнює значенню блока як 4-бітового цілого додатного числа. Тепер стає зрозумілим розмір таблиці замін: кількість рядків у ній дорівнює кількості 4-бітових елементів у 32-бітовому блоці даних, тобто восьми, а кількість стовпців дорівнює кількості різних значень 4-бітових блоків даних, тобто 2^4 або шістнадцяти.

Крок 3. *Циклічний зсув на 11 бітів уліво*. Результат попереднього кроку зсувають циклічно на 11 бітів у бік старших розрядів і передають на наступний крок. На схемі алгоритму це позначено символом $R_{\overleftarrow{11}}$ (рис. 2.18).

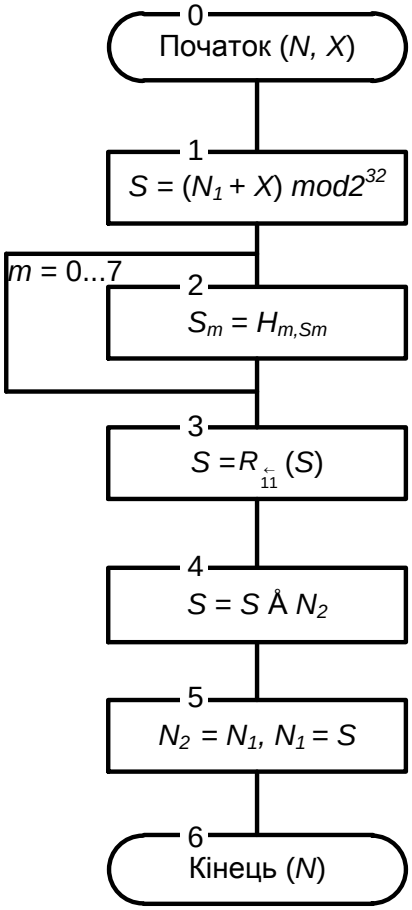


Рис. 2.18. **Схема основного кроку криптоперетворення алгоритму ГОСТ 28147-89**

Крок 4. *Побітове додавання:* значення, визначене на кроці 3, побітово додають за модулем 2 до старшої половини блока.

Крок 5. *Зсув за ланцюжком:* обчислені значення старшого та молодшого півблоків міняють місцями.

Крок 6. Визначене значення перетвореного блока повертають як результат виконання алгоритму основного кроку криптоперетворення.

ГОСТ 28147-89 передбачає використання перетворення у трьох режимах:

- режимі простої заміни;
- режимі гамування;
- режимі гамування зі зворотним зв'язком;
- одному додатковому режимі генерування імітовставки.

2.4. Практичне виконання шифрування

1. Виберіть алгоритм і режим шифрування (рис. 2.19).

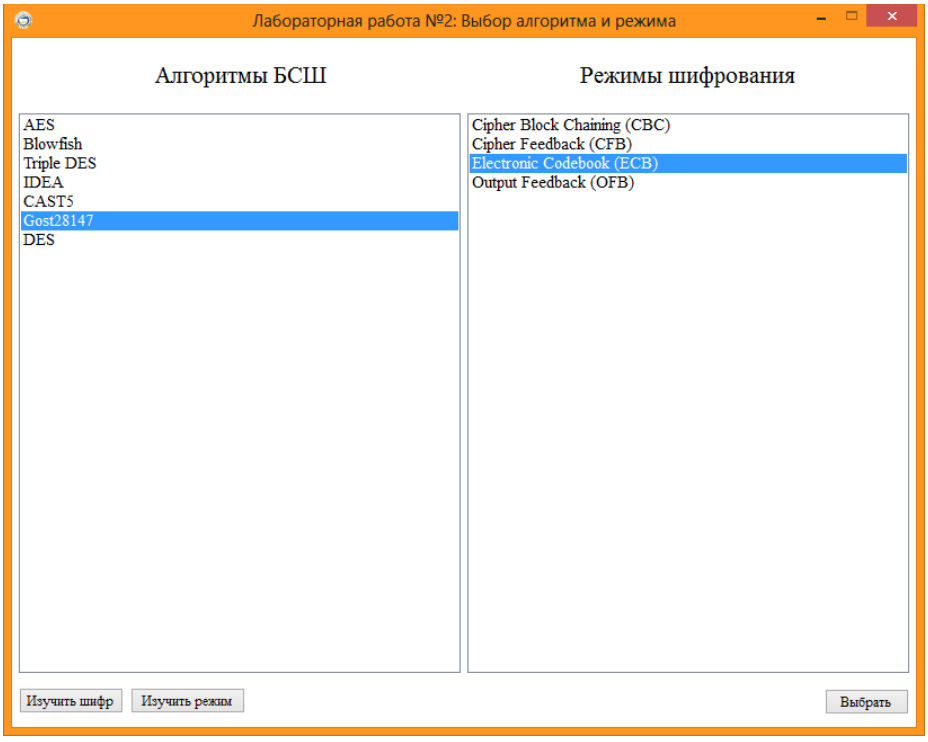


Рис. 2.19. **Вибір алгоритму та режиму шифрування**

2. Натисніть кнопку "Редактировать текст".

3. Уведіть текст, у якому повторюються перші чотири блоки відкритого тексту. Таким відкритим текстом виберіть ПІБ студента.

Приклад: ПІБ – КнигаАлександрАлексеевич (рис. 2.20). У шифруванні алгоритмом Gost на режимі ECB довжина блока перетворення дорівнює 64 бітам. У форматі ASCII кожній літері відповідає один байт. Один блок відкритого тексту дорівнює 8 байтам (табл. 2.3).

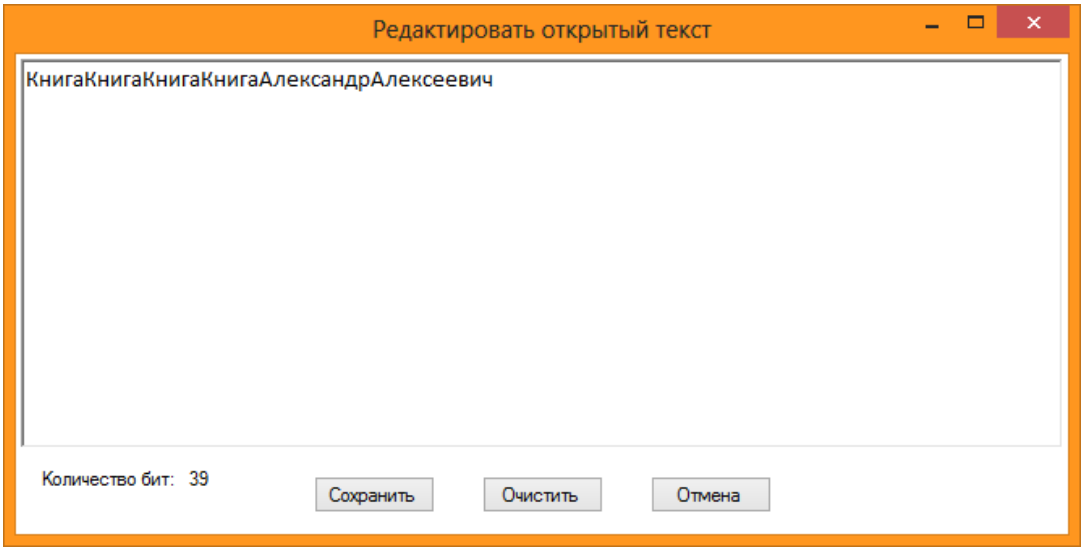


Рис. 2.20. **Додавання відкритого тексту**

4. Перегляньте результат у бінарному поданні (рис. 2.21), натиснувши на кнопку "Редактировать текст в бинарном виде" (рис. 2.22).

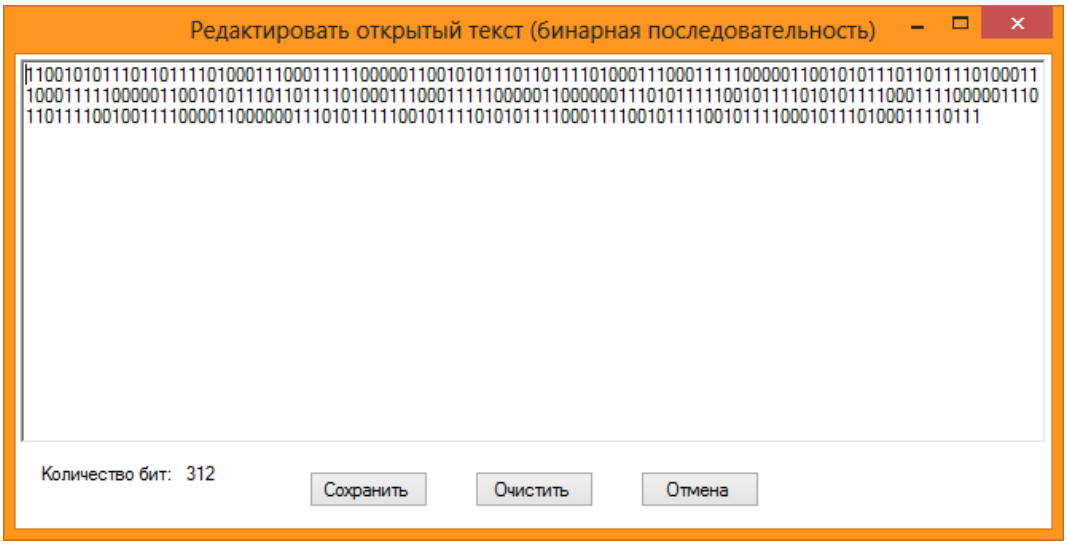


Рис. 2.21. **Перегляд бітового коду відкритого тексту**

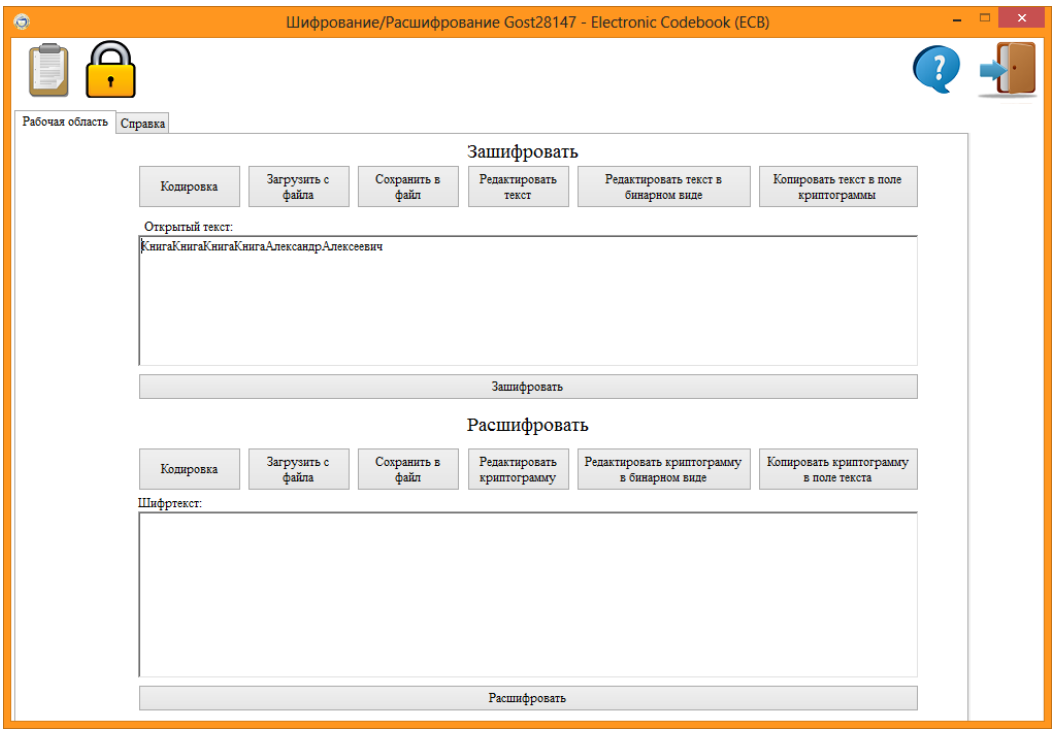


Рис. 2.22. **Перегляд результату додавання відкритого тексту на головному екрані**

Таблиця 2.3

Байти кожного слова відкритого тексту

Слово	Книга	Книга	Книга	Книга	Александр	Алексеевич
Бітове значення в ASCII	11001010 11101101 11101000 11100011 11100000	11001010 11101101 11101000 11100011 11100000	11001010 11101101 11101000 11100011 11100000	11001010 11101101 11101000 11100011 11100000	11000000 11101011 11100101 11101010 11110001 11100000 11101101 11100100 11110000	11000000 11101011 11100101 11101010 11110001 11100101 11100101 11100010 11101000 11110111

Довжина кожного блока має бути 8 байтів (64 біти), чого не дотримано в п'ятому блоці (табл. 2.4).

5. Натисніть кнопку "Зашифровать" у головному вікні. Буде запропоновано ввести фразу-пароль і задати крок генератора, щоб створити вектор ініціалізації (рис. 2.23).

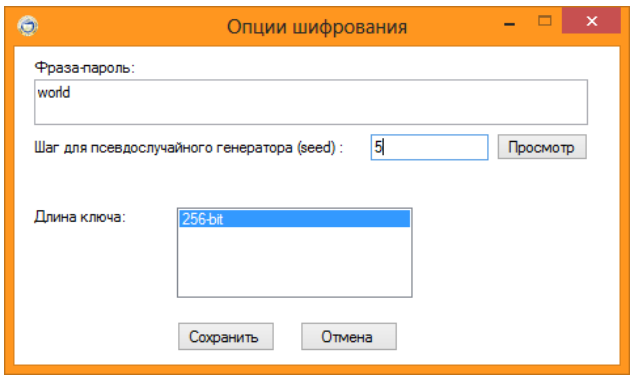


Рис. 2.23. **Опції шифрування**

Таблиця 2.4

Байти кожного блока відкритого тексту

Номер блока	1	2	3	4	5
Бітове значення в ASCII	11001010 11101101 11101000 11100011 11100000 11001010 11101101 11101000	11100011 11100000 11001010 11101101 11101000 11100011 11100000 11001010	11101101 11101000 11100011 11100000 11000000 11101011 11100101 11101010	11110001 11100000 11101101 11100100 11110000 11000000 11101011 11100101	11101010 11110001 11100101 11100101 11100010 11101000 11110111

Оскільки кількість байтів відкритого тексту не є кратною 8, буде виведено таке повідомлення про помилку (рис. 2.24).

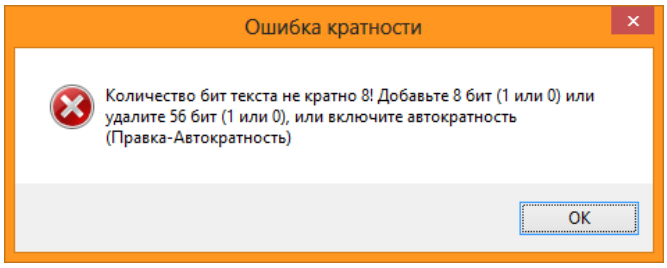


Рис. 2.24. **Помилка кратності**

Вирішити проблему можна, дописавши вручну відсутню кількість байтів або включивши автоматичну кратність у налаштуваннях.

6. Вирішіть проблему із кратністю. Один із варіантів результату вирішення показано в табл. 2.5.

Таблиця 2.5

Порівняння останнього блока до і після редагування

Останній блок до редагування	Останній блок після редагування
11100010	11100010
11101000	11101000
11110111	11110111
	11111111
	11111111
	11111111
	11111111
	11111111

Після успішного виконання алгоритму у блоці "Расшифровать" з'явиться зашифрований текст (рис. 2.25) та зашифрований текст у бінарному вигляді (рис. 2.26).

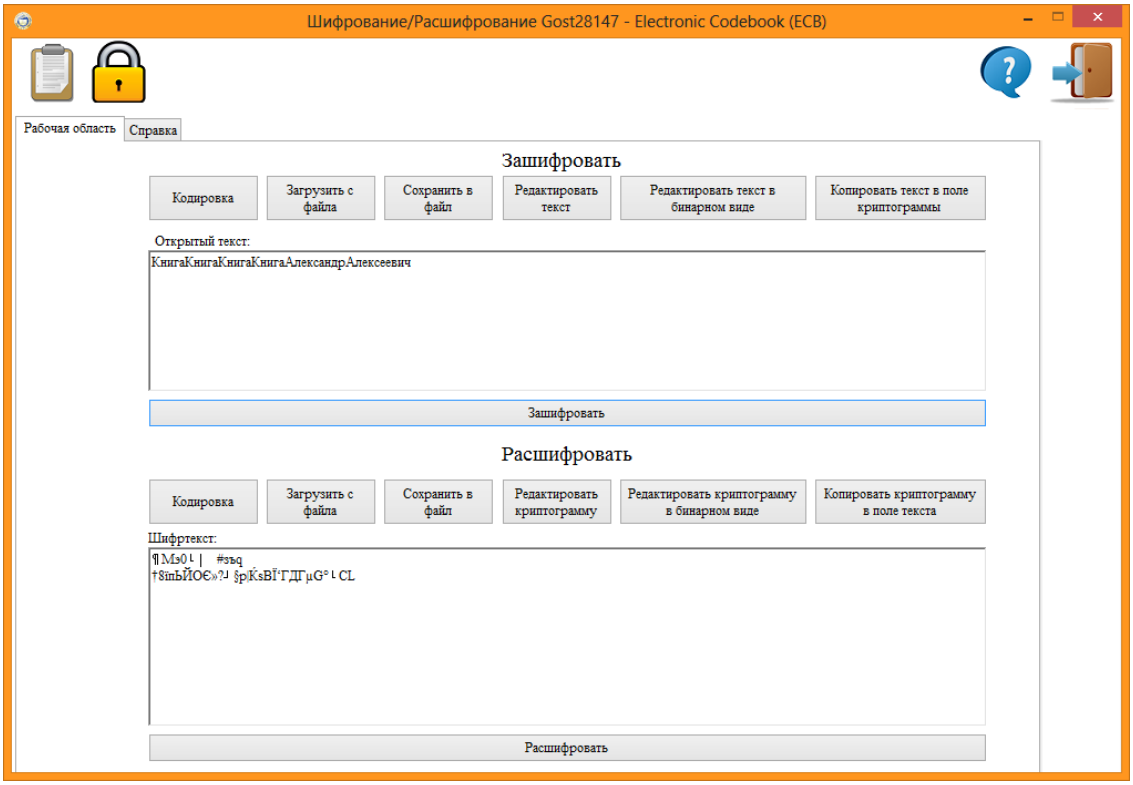


Рис. 2.25. **Зашифрований текст на головному екрані**

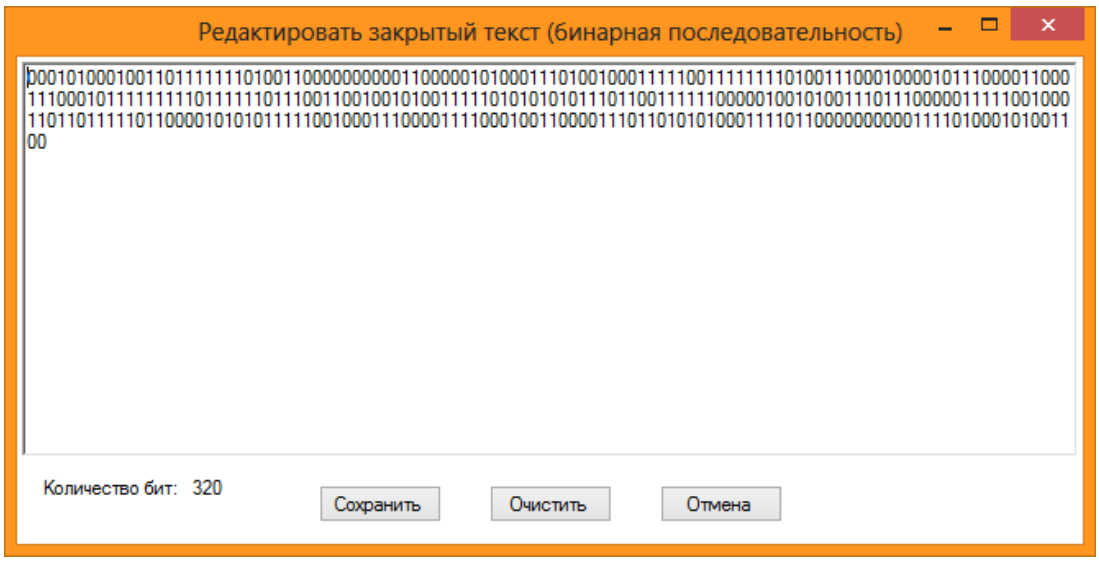


Рис. 2.26. **Бінарний варіант зашифрованого тексту**

7. Проаналізуйте визначений шифротекст, порівняйте його зі схемою використовуваного режиму. У режимі ECB зверніть увагу на повторювані блоки режиму (табл. 2.6).

Таблиця 2.6

Порівняння відкритого та зашифрованого текстів

Тип	До шифрування	Після шифрування
Бінарний код	11001010111011011111010001110000111	00010100010011011111110100110000000
	11000001100101011101101111010001 1	00011000001010001110100100011111001
	10001111100000110010101110110111 1	11111110100111000100001011100001100
	01000111000111110000011001010111 0	01110001011111111101111110111001100
	11011110100011100011111000001 1000	10010100111110101010101110110011111
	00011101011111001011110101011110 0	10000010010100111011100000111110010
	01111000001110110111100100111000 0	00110110111110110000101010111110010
	01100000011101011111001011101010	00111000011110001001100001110110101
	11110001111001011110010111100010 1	01000111101100000000001111010001010
	110100011110111	01100
Текст	КнигаКнигаКнигаКнигаАлександрАлексеевич	Мэ0#зъq†8їпЬЙОЄ"?§р|ЌsВЇ'ГДГµG°CL

8. Розшифруйте визначений шифротекст. Результат показано на рис. 2.27.

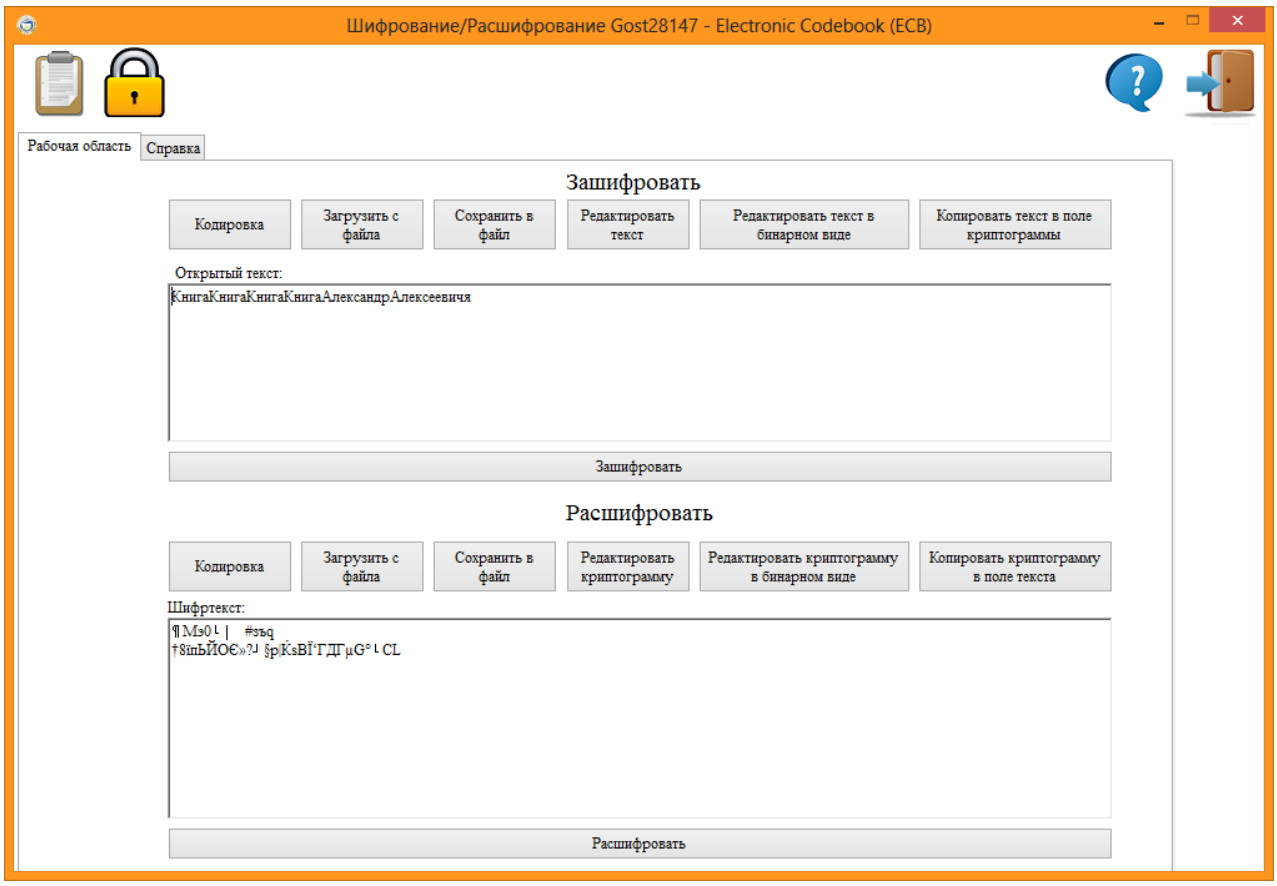


Рис. 2.27. **Розшифрування шифротексту на головному екрані**

9. Для здійснення аналізу поширення помилок унесіть зміни в перший біт другого блока шифротексту та розшифруйте його (рис. 2.28 і 2.29).

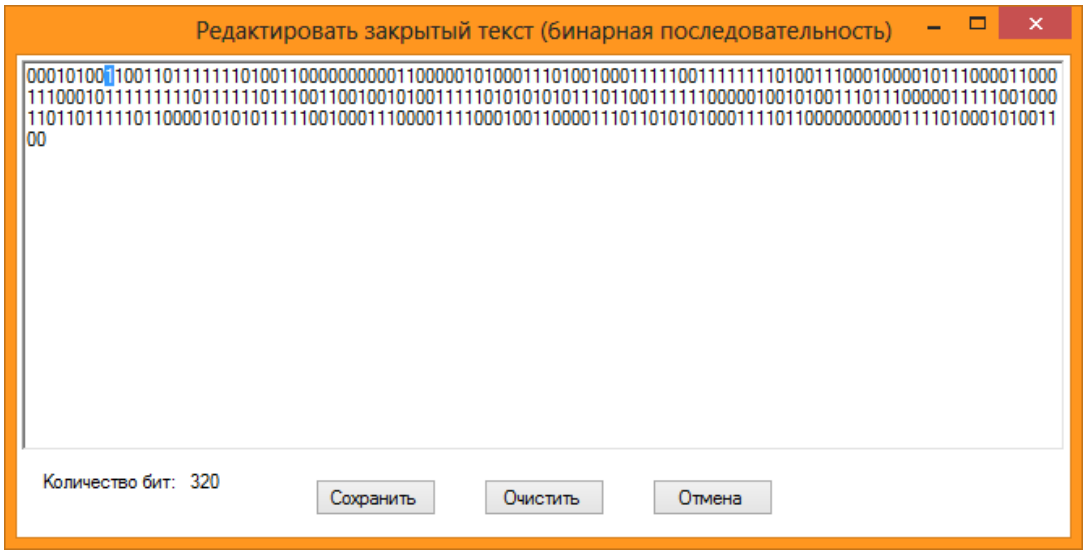


Рис. 2.28. **Унесення помилки в перший біт другого байта зашифрованого тексту**

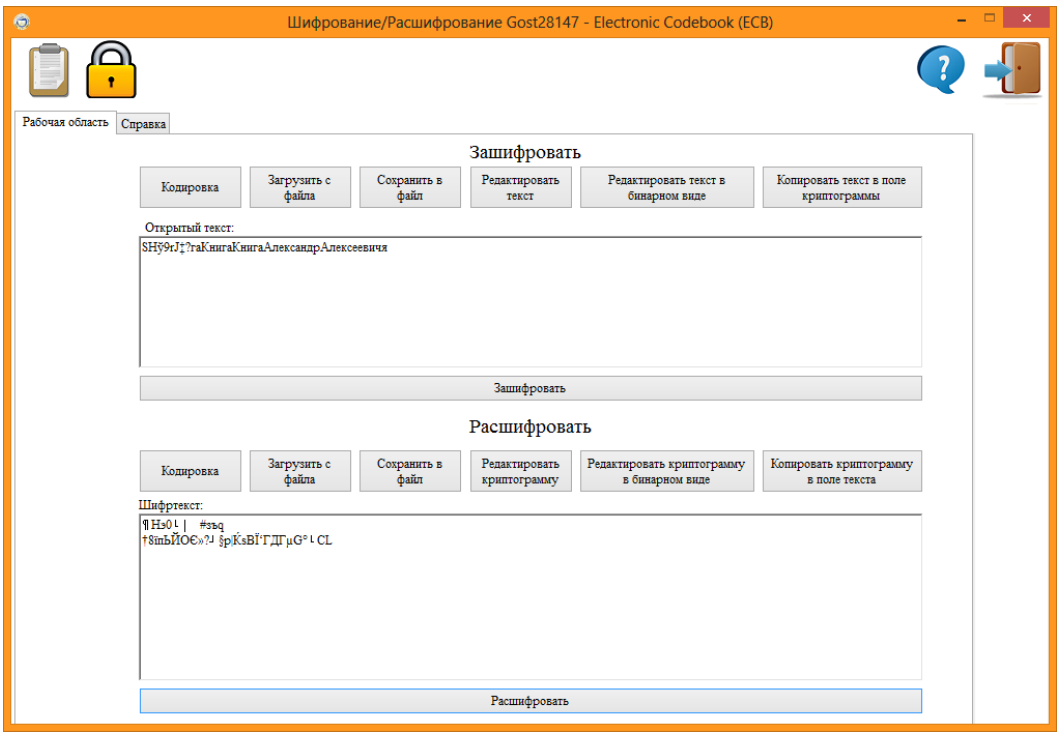


Рис. 2.29. **Результат розшифрування зашифрованого тексту з помилкою**

10. Проаналізуйте поширення помилки, порівняйте результати зі схемою режиму (табл. 2.7).

Таблиця 2.7

Порівняння відкритого тексту з помилкою та без помилки

Типи	Відкритий текст.	Розшифрування шифротексту з помилкою в першому біті другого байта
Бінарний код	11001010111011011111010000111000111	01010011010010001010001000111001011
	11000001100101011101101111010011	10010101000111000011100111111111000
	10001111100000110010101110110111	11111000001100101011101101111010001
	01000111000111110000011001010111	11000111110000011001010111011011110
	11011110100011100011111000001100	10001110001111100000110000001110101
	00011101011111001011110101011110	11110010111101010111100011110000011
	01111000001110110111100100111000	10110111100100111100001100000011101
	01100000011101011111001011101010	01111100101111010101111000111100101
	11110001111001011110010111100101	11100101111000101110100011110111111
	110100011110111	11111
Текст	КнигаКнигаКнигаКнигаАлександрАлексеевича	SHў9rJ‡? гаКнигаКнигаАлександрАлексеевичя

Виконайте ці кроки для всіх режимів шифрування та проаналізуйте результати. У звіт додайте скриншоти відкритого тексту, шифротексту, шифротексту з помилкою та відкритого тексту.

2.5. Завдання до лабораторної роботи

1. Виконайте аналіз режимів шифрування ECB, CTS, CBC, CFB, OFB із погляду поширення помилок шифрування (табл. 2.8). У режимах використайте шифри, відповідно до варіанта.

Таблиця 2.8

Варіанти індивідуального завдання

Варіанти	Шифри	Блоки для зміни
1	Blowfish, Gost	0, 1, передостанній, останній
2	Des, Gost	0, 1, передостанній, останній
3	Square, Gost	0, 1, передостанній, останній
4	Gost, rijndael	0, 1, передостанній, останній
5	Gost, twofish	0, 1, передостанній, останній
6	Gost, Idea	0, 1, передостанній, останній
7	Gost, Mars	0, 1, передостанній, останній
8	Gost, rc6	0, 1, передостанній, останній
9	Cast 256, Gost	0, 1, передостанній, останній
10	RC5, Gost	0, 1, передостанній, останній

2.6. Контрольні запитання

1. Які ви знаєте принципи симетричного шифрування?
2. Яка структура блокового симетричного шифру?
3. Які є рівні стійкості блокових симетричних шифрів?
4. Що таке "схема розгортання ключів"?
5. Охарактеризуйте ланцюг Фейстеля, його переваги та недоліки.
6. Які ви знаєте потокові режими шифрування?
7. Що таке "цілісність" та "ідентифікатор повідомлення"?
8. Що називають набиванням повідомлення?
9. Розкажіть про характеристики режимів CBC і CTS.
10. Охарактеризуйте поширення помилки в режимі CFB.

Лабораторна робота 3
Асиметричні криптосистеми

3.1. Мета

Мета цієї лабораторної роботи полягає в набутті студентами навичок у використанні механізмів асиметричного шифрування для забезпечення конфіденційності повідомлень. Як приклад використовують асиметричні алгоритми RSA та ДСТУ ISO/IEC 15948-3.

3.2. Рекомендації до підготовки до виконання

Для виконання лабораторної роботи студенту необхідно орієнтуватися в основних поняттях і визначеннях із питань конфіденційності; вивчити принципи роботи асиметричних криптоперетворень; ознайомитися з основними принципами роботи алгоритмів RSA та ДСТУ ISO/IEC 15948-3.

3.3. Загальні теоретичні положення

Алгоритм RSA

Назва алгоритму походить від перших літер прізвищ авторів: Рональда Рівеста (Ron Rivest), Аді Шаміра (Adi Shamir) і Леонарда Адлемана (Leonard Adleman).

Криптосистема RSA використовує одностороннюю функцію утворення добутку двох великих простих цілих чисел, що значно простіше, ніж розкладання великого цілого числа на прості множники. Зрозуміло, що принципово це можна зробити, однак витрати ресурсів (часу або обчислювальної потужності комп'ютерів) будуть не меншими, ніж просте суцільне перебирання всього ключового масиву.

В алгоритмі використовують два ключі: публічний та приватний.

Публічний ключ може бути опубліковано в довіднику поряд з іменем користувача. У результаті будь-хто може зашифрувати з його допомогою свій лист і надіслати зашифровану інформацію власнику відповідного приватного ключа.

Розшифрувати надіслане повідомлення зможе тільки той, у кого є *приватний ключ*. Розшифрувати це повідомлення за допомогою публічного ключа вкрай важко, це потребує надзвичайних обчислювальних ресурсів.

У цьому процесі відбуваються такі перетворення:

- *повідомлення + публічний ключ користувача А = шифротекст;*
- *шифротекст + приватний ключ користувача А = повідомлення.*

Таким чином, кожен може надіслати користувачеві *А* секретну інформацію, скориставшись його публічним ключем. Але тільки користувач *А* у змозі розшифрувати повідомлення, оскільки лише в нього є відповідний приватний ключ.

Причина працездатності таких криптосистем полягає в однобічному математичної зв'язку, що є між двома ключами, за якого інформація про публічний ключ не допомагає відновити приватний. Але володіння приватним ключем забезпечує можливість розшифровувати повідомлення, зашифровані публічним ключем. На перший погляд, такий зв'язок здається дивним, та для його розуміння потрібен певний час і розумові зусилля.

Опис алгоритму шифрування. Першим етапом будь-якого асиметричного алгоритму є створення двох ключів – публічного та приватного, та поширення відкритого ключа "по всьому світу". Для алгоритму *RSA* етап створення ключів складається з таких операцій:

1) вибору двох простих чисел p і q;

2) обчислення їхнього добутку $n = p \times q$;

3) обчислення функції Ейлера $\varphi(n) = (p - 1)(q - 1)$;

4) вибору довільного числа e ($e < n$) – такого, щоб *НСД* $(e, \varphi(n)) = 1$, тобто e має бути взаємно простим із числом $\varphi(n)$;

5) обчислення числа $d = e^{-1} \bmod \varphi(n)$;

6) визначення публічного ключа $K_U = \{e, n\}$;

7) визначення приватного ключа $K_R = \{d, n\}$;

8) шифрування. Незашифрований текст має бути $M < n$; $C = M^e \pmod n$;

9) розшифрування. Зашифрований текст $C : M = C^d \pmod n$.

Приклад роботи алгоритму RSA:

- виберіть два простих числа: $p = 7$, $q = 17$;
- обчисліть $n = p \times q = 7 \times 17 = 119$;
- обчисліть $\varphi(n) = (p - 1)\,(q - 1) = 96$;
- виберіть e так, щоб воно було взаємно простим із $\varphi(n) = 96$ і меншим за $\varphi(n)$: $e = 5$;
- визначте d так, щоб $d = e^{-1} \bmod 96$: $d = 77$, оскільки $77 \times 5 = 385 = 4 \times 96 + 1$.

Результивні ключі: публічний – $K_U = \{5, 119\}$, приватний – $K_R = \{77, 119\}$.

Нехай необхідно зашифрувати повідомлення $M = 19$: $19^5 \bmod 119 = 66$.

Для розшифрування обчисліть 66^{77} $(mod\ 119) = 19$.

На практиці загальнодоступні ключі можуть розміщувати у спеціальній БД. За потреби послати партнеру зашифроване повідомлення можна зробити спочатку запит його відкритого ключа. Отримавши його, можна запустити програму шифрування, а результат її роботи надіслати адресатові.

На рис. 3.1 проілюстровано принцип шифрування/розшифрування методом RSA.

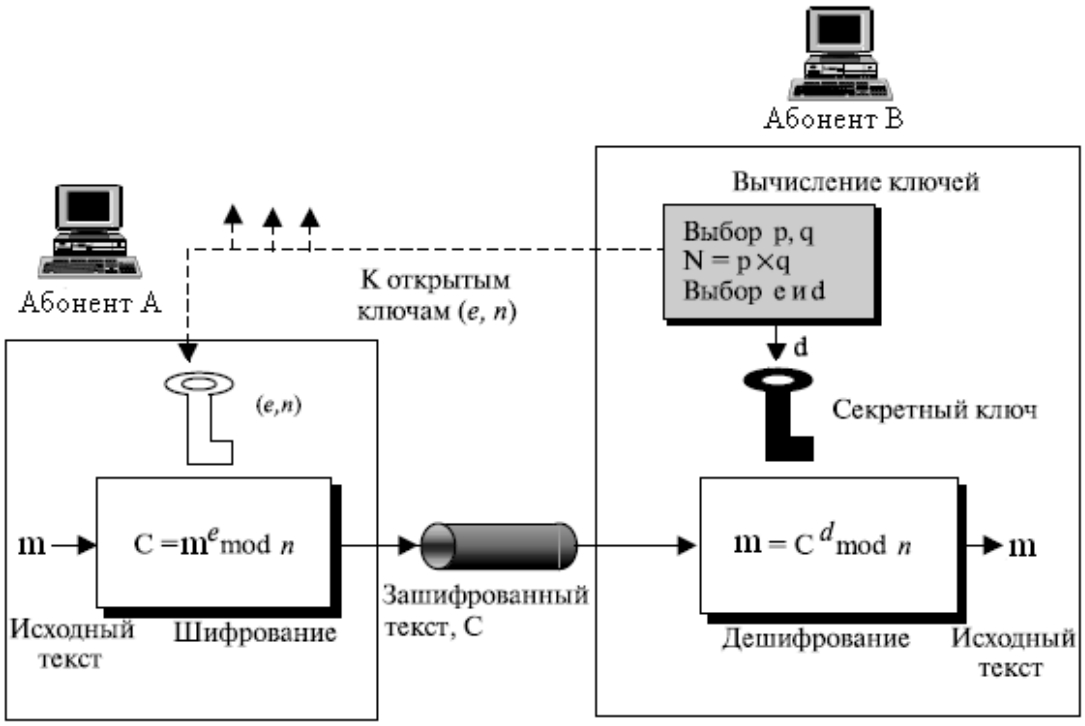


Рис. 3.1. **Криптосистема RSA**

Алгоритм обміну ключа Діффі – Геллмана

Перша публікація цього алгоритму погодження ключа з'явилася у статті У. Діффі та М. Геллмана, де введено основні поняття криптографії з публічним ключем і в загальних рисах описано алгоритм погодження ключа Діффі – Геллмана. Мета алгоритму полягає в тому, щоб два учасники могли безпечно погодити ключ, який надалі можна використовувати в будь-якому алгоритмі симетричного шифрування. Сам алгоритм Діффі – Геллмана можна застосовувати тільки для обміну ключами.

Алгоритм засновано на проблемі обчислень дискретних логарифмів. Дискретний логарифм визначають таким чином. Уводять поняття примітивного кореня простого числа *Q* як числа, чиї степені створюють усі цілі числа від *1* до *Q - 1*. Це означає, що якщо *A* є примітивним коренем простого числа *Q*, то числа $A \bmod Q, A^2 \bmod Q, \ldots, A^{Q-1} \bmod Q$ різні та складаються із цілих чисел від *1* до *Q - 1* із деякими перестановками. У цьому разі для будь-якого цілого $Y < Q$ і примітивного кореня *A* простого числа *Q* можна знайти єдину експоненту *X* – таку, що $Y = A^X \bmod Q$, *де* $0 \le X \le (Q - 1)$.

Експоненту *X* називають *дискретним логарифмом*, або індексом *Y*, за модулем *Q*. Це позначають як $ind_{A,Q}(Y)$.

Тепер слід описати алгоритм обміну ключів Діффі – Геллмана (табл. 3.1).

Таблиця 3.1

Алгоритм обміну ключів Діффі – Геллмана

Загальновідомі елементи	
Q	просте число
P	$P < Q$ і *P* є примітивним коренем *Q*
Створення пари ключів користувачем I	
Вибір випадкового числа X_i (закритий ключ)	$X_i < Q$
Обчислення числа Y_i (відкритий ключ)	$Y_i = P^{Xi} \bmod Q$
Створення відкритого ключа користувачем *J*	
Вибір випадкового числа X_j (закритий ключ)	$X_j < Q$
Обчислення випадкового числа Y_j (відкритий ключ)	$Y_j = P^{Xj} \bmod Q$
Створення загального секретного ключа користувачем *I*	
$K = (Y_j)^{Xi} \bmod Q$	
Створення загального секретного ключа користувачем *J*	
$K = (Y_i)^{Xj} \bmod Q$	

Передбачають, що є два відомих усім числа: просте число *Q* і ціле *P*, яке є примітивним коренем *Q*. Припустіть, що користувачі *I* та *J* хочуть обмінятися ключем для алгоритму симетричного шифрування. Користувач *I* вибирає випадкове число $X_i < Q$ і обчислює $Y_i = P^{Xi} \bmod Q$. Аналогічно користувач *J* незалежно вибирає випадкове ціле число $X_j < Q$ і обчислює $Y_j = P^{Xj} \bmod Q$. Сторони обмінюються значеннями *Y* та тримають у секреті значення *X*. Тепер користувач *I* обчислює ключ як $K = (Y_j)^{Xi} \bmod Q$,

і користувач *J* обчислює ключ як $K = (Y_i)^{Xj} \bmod Q$. У результаті обидва знайдуть одне й те саме значення:

$$K = (Y_j)^{Xi} \bmod Q = (P^{Xj} \bmod Q)^{Xi} \bmod Q = (P^{Xj})^{Xi} \bmod Q = P^{XjXi} \bmod Q =$$
$$= (P^{Xj})^{X} \bmod Q = (P^{Xi} \bmod Q)^{Xj} \bmod Q = (Y_i)^{Xj} \bmod Q.$$

Таким чином, дві сторони обмінялися секретним ключем. Оскільки X_i і X_j є закритими, противник може знайти тільки такі значення: *Q, P,* Y_i й Y_j. Для обчислення ключа він має розв'язати задачу дискретного логарифмування, тобто обчислити:

$$X_j = ind_{a,q}(Y_j). \qquad (3.1)$$

Безпека обміну ключа в алгоритмі Діффі – Геллмана випливає з того факту, що, хоча відносно легко обчислити експоненти за модулем простого числа, але дуже важко обчислити дискретні логарифми. Для великих простих чисел задачу вважають нерозв'язною.

Слід зауважити, що цей алгоритм вразливий для атак типу *man-in-the-middle*. Якщо противник може здійснити активну атаку, тобто має можливість не тільки перехоплювати повідомлення, але й замінювати їх іншими, він може перехопити відкриті ключі учасників Y_i і Y_j, створити свою пару відкритого та закритого ключів ($X_{оп}$, $Y_{оп}$) і послати кожному з учасників свій відкритий ключ. Після цього кожен учасник вирахує ключ, який буде спільним із противником, а не з іншим учасником. Якщо немає контролю за цілісністю, то учасники не зможуть виявити подібну підміну.

Протоколи забезпечення конфіденційності й автентичності

Перший варіант – передавання повідомлень із відкритим ключем абонента *B* (KU_B) – подано на рис. 3.2.

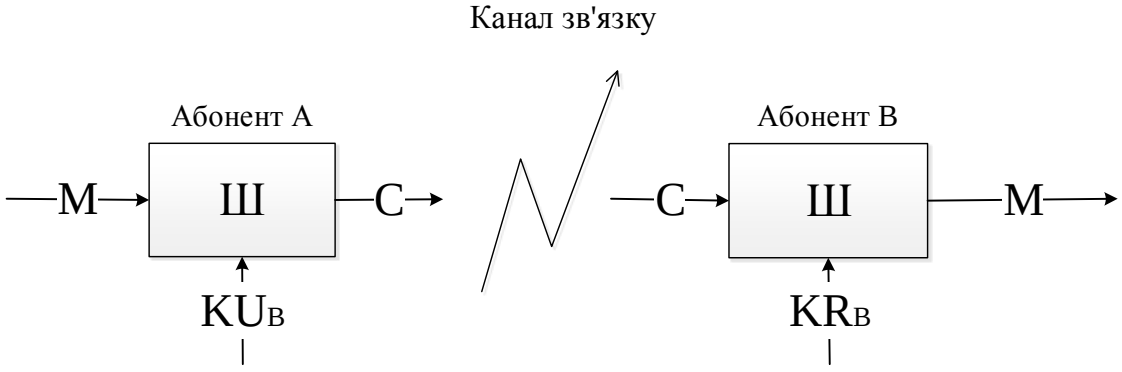


Рис. 3.2. **Забезпечення конфіденційності**

Таким чином забезпечено конфіденційність повідомлення. Недоліком схеми є нездатність забезпечення автентичності.

Другий варіант – використання абонентом *A* під час відправлення повідомлення свого приватного ключа (KR_A) – подано на рис. 3.3.

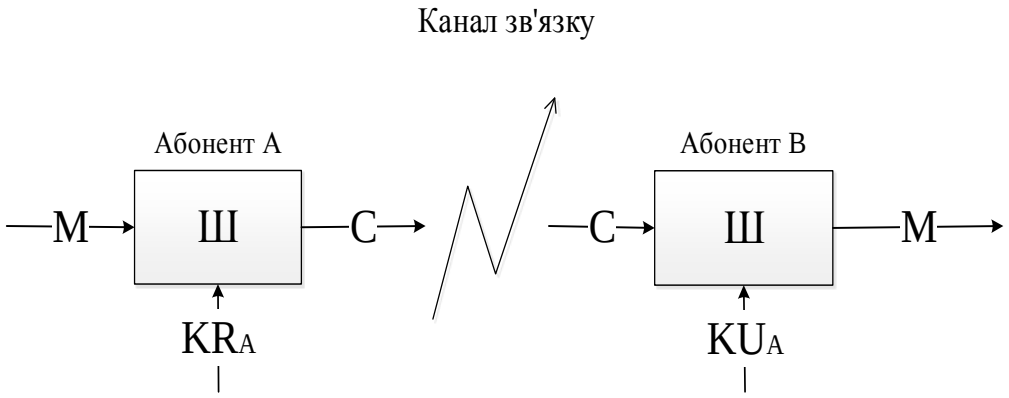


Рис. 3.3. **Забезпечення автентичності**

Водночас забезпечено автентичність і цілісність повідомлення. Недоліком є можливість для будь-якого абонента використати публічний ключ KU_A, щоб перевірити підпис.

Третій варіант – використання абонентами своїх ключів для обміну повідомленнями – зображено на рис. 3.4. Тут забезпечено конфіденційність, цілісність та автентичність.

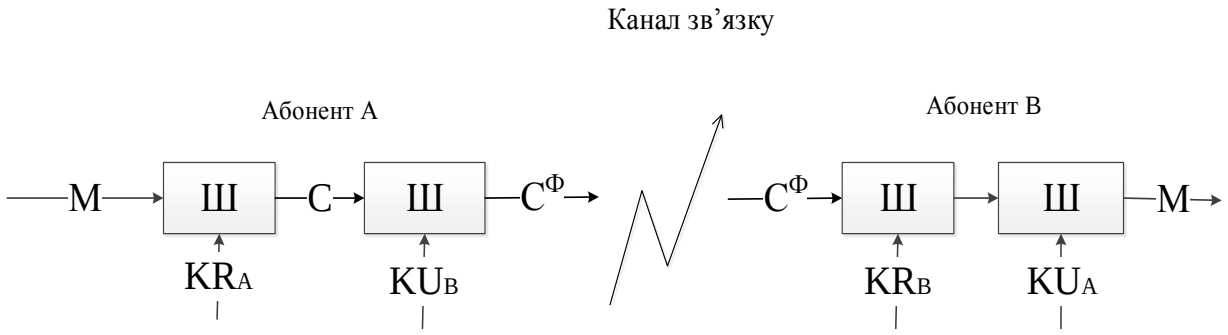


Рис. 3.4. **Забезпечення конфіденційності й автентичності**

3.4. Практичне виконання лабораторної роботи

Алгоритм RSA

1. Щоб зашифрувати повідомлення за допомогою алгоритму RSA, необхідно сформувати приватний та публічний ключі системи. Для цього необхідно ввести розрядність простих чисел *p* і *q*, на якій ґрунтується криптостійкість системи (рис. 3.5).

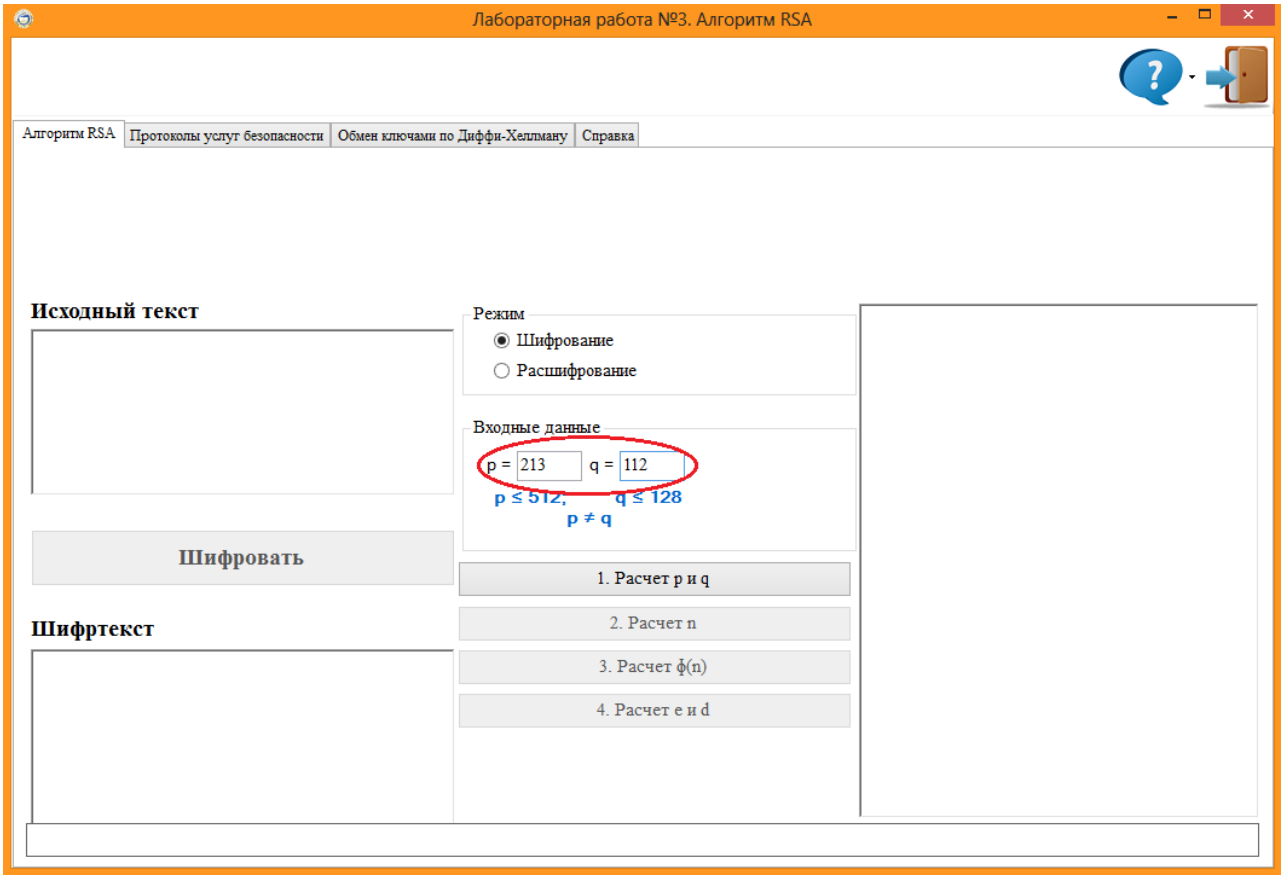


Рис. 3.5. **Заповнення вхідних даних**

2. Тепер необхідно виконати таку послідовність дій:

- розрахувати прості числа *p* і *q*;
- розрахувати *n*;
- розрахувати функцію Ейлера $\varphi(n)$;
- розрахувати *e* і *d* – публічний та приватний ключі.

Інтерфейс програми надає можливість виконати всі дії у правильному порядку (рис. 3.6).

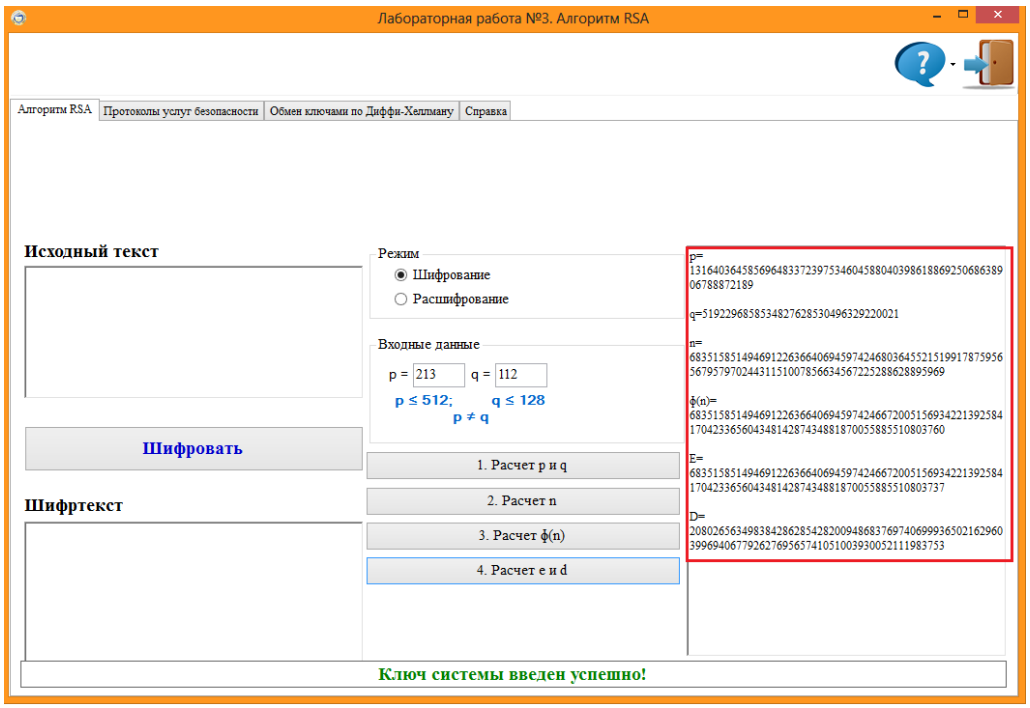


Рис. 3.6. **Розрахунок необхідних змінних**

3. Після успішного розрахунку ключів системи, можна розпочати шифрування повідомлень.

Для цього необхідно ввести текст повідомлення та натиснути кнопку "Шифровать" (рис. 3.7).

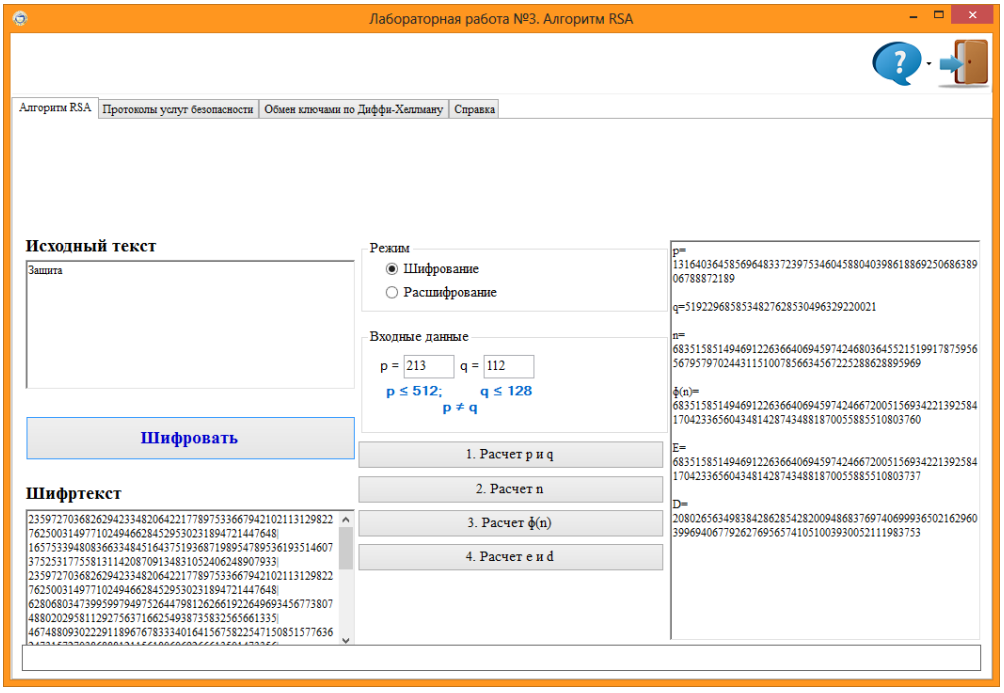


Рис. 3.7. **Зашифрування тексту**

4. Щоб розшифрувати шифротекст, необхідно:

• установити режим "Расшифрование";

• скопіювати шифротекст у поле "Шифротекст" або скористатися кнопкою "Копировать", яка зробить це автоматично;

• натиснути кнопку "Расшифровать" (рис. 3.8).

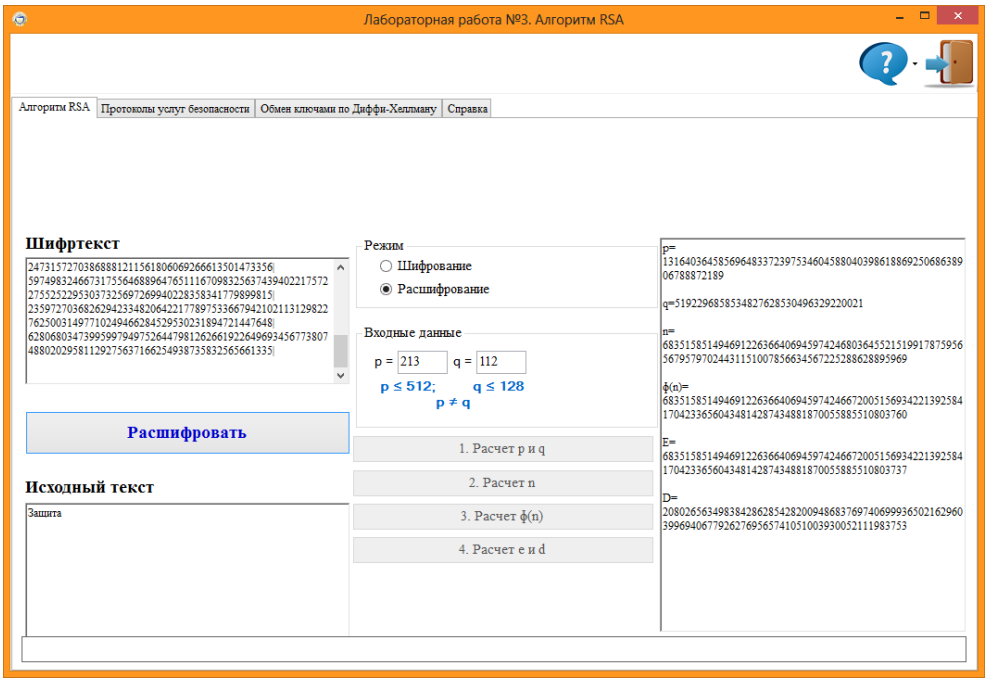


Рис. 3.8. **Розшифрування криптограми**

Алгоритм обміну ключа Діффі – Геллмана

Для роботи з алгоритмом Діффі – Геллмана необхідно виконати шифрування даних за допомогою алгоритму RSA, виконавши всі необхідні налаштування:

1. Увести секретний ключ абонента *A* (він має бути простим числом).
2. Увести секретний ключ абонента *B* (він має бути простим числом).
3. Натиснути кнопку "Подтвердить ввод *x*".
4. Натиснути кнопку "Подтвердить ввод *y*".
5. Натиснути кнопку "Сформировать открытый ключ X_a".
6. Натиснути кнопку "Сформировать открытый ключ Y_b".
7. Натиснути кнопку "Сформировать общий секретный ключ K_a".
8. Натиснути кнопку "Сформировать общий секретный ключ K_b".
9. Якщо сформовані загальні секретні ключі абонентів збігаються, то обмін ключами за алгоритмом Діффі – Геллмана було здійснено успішно (рис. 3.9).

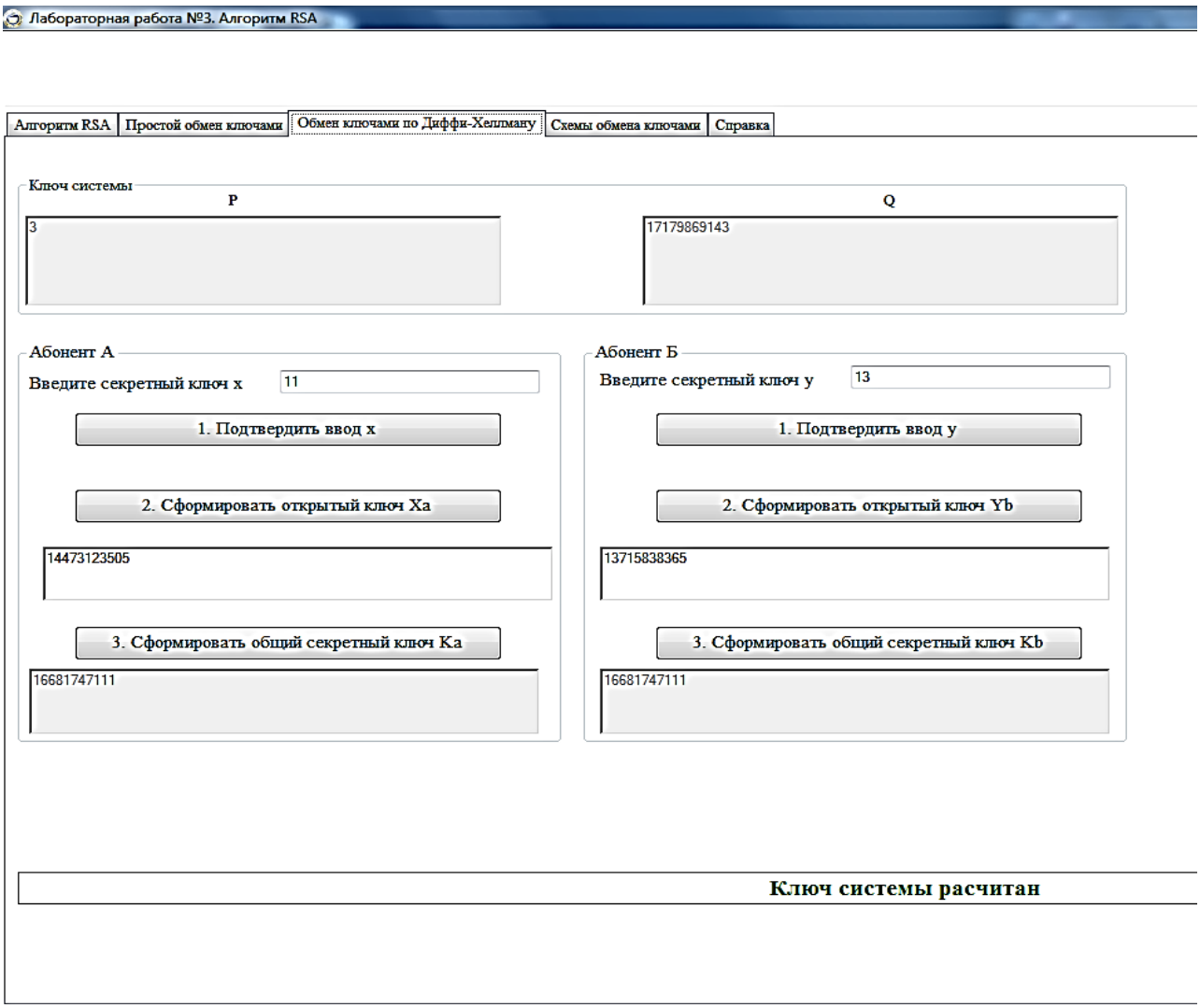


Рис. 3.9. **Обмін ключами за алгоритмом Діффі – Геллмана**

Протоколи забезпечення конфіденційності й автентичності

Для забезпечення автентичності та конфіденційності, необхідно виконати шифрування даних за допомогою алгоритму RSA, виконавши всі необхідні налаштування.

Забезпечення автентичності:

1. Виберіть режим "Аутентичность".
2. Уведіть вихідний текст.
3. Натисніть кнопку "Ш" (абонента *А*).
4. Натисніть кнопку "Ш" (абонента *Б*).
5. Розшифрований шифротекст має збігатися з вихідним текстом (рис. 3.10).

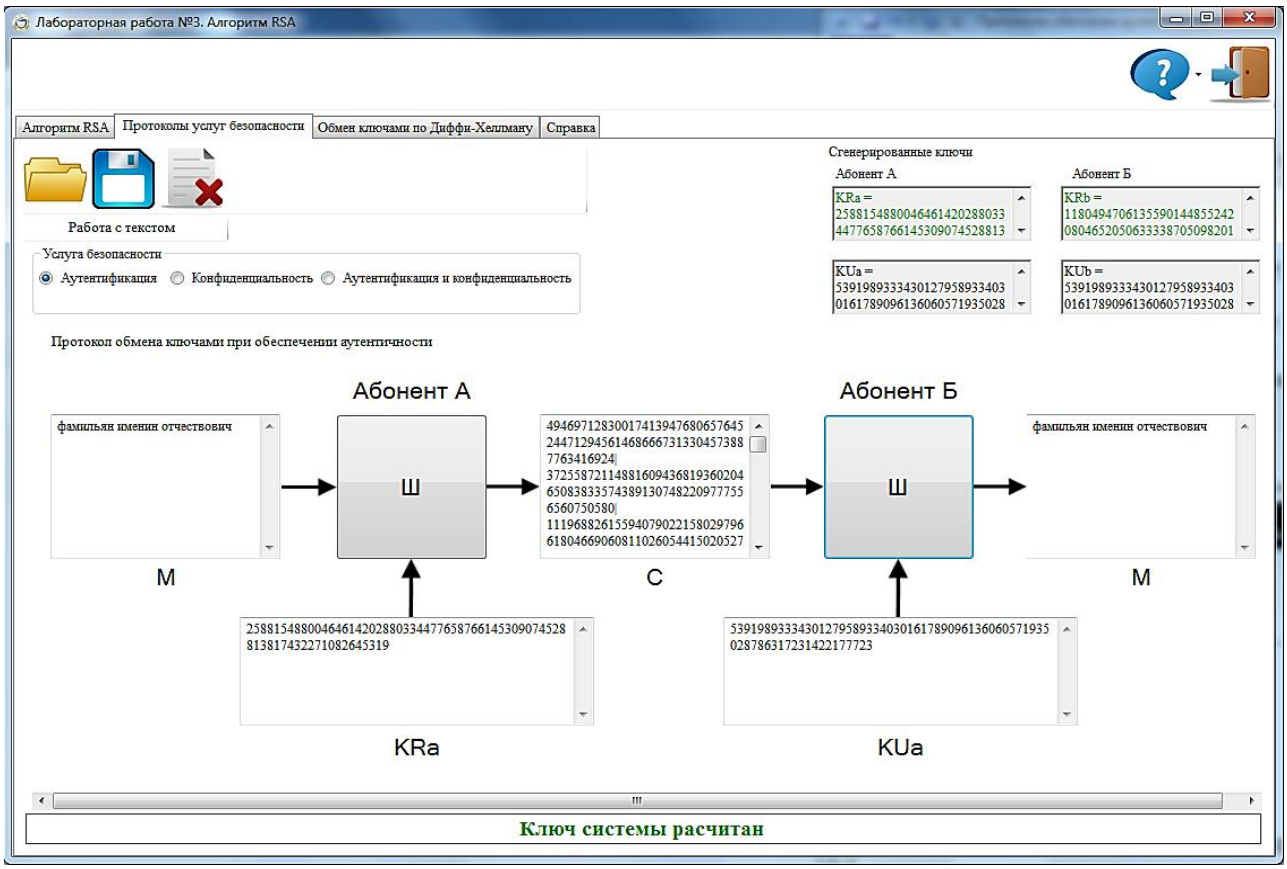


Рис. 3.10. **Розшифрування шифротексту**

Забезпечення конфіденційності:

1. Виберіть режим "Конфиденцальность".
2. Уведіть вихідний текст.
3. Натисніть кнопку "Ш" (абонента А).
4. Натисніть кнопку "Ш" (абонента Б).
5. Розшифрований шифротекст має збігатися з вихідним текстом (рис. 3.11).

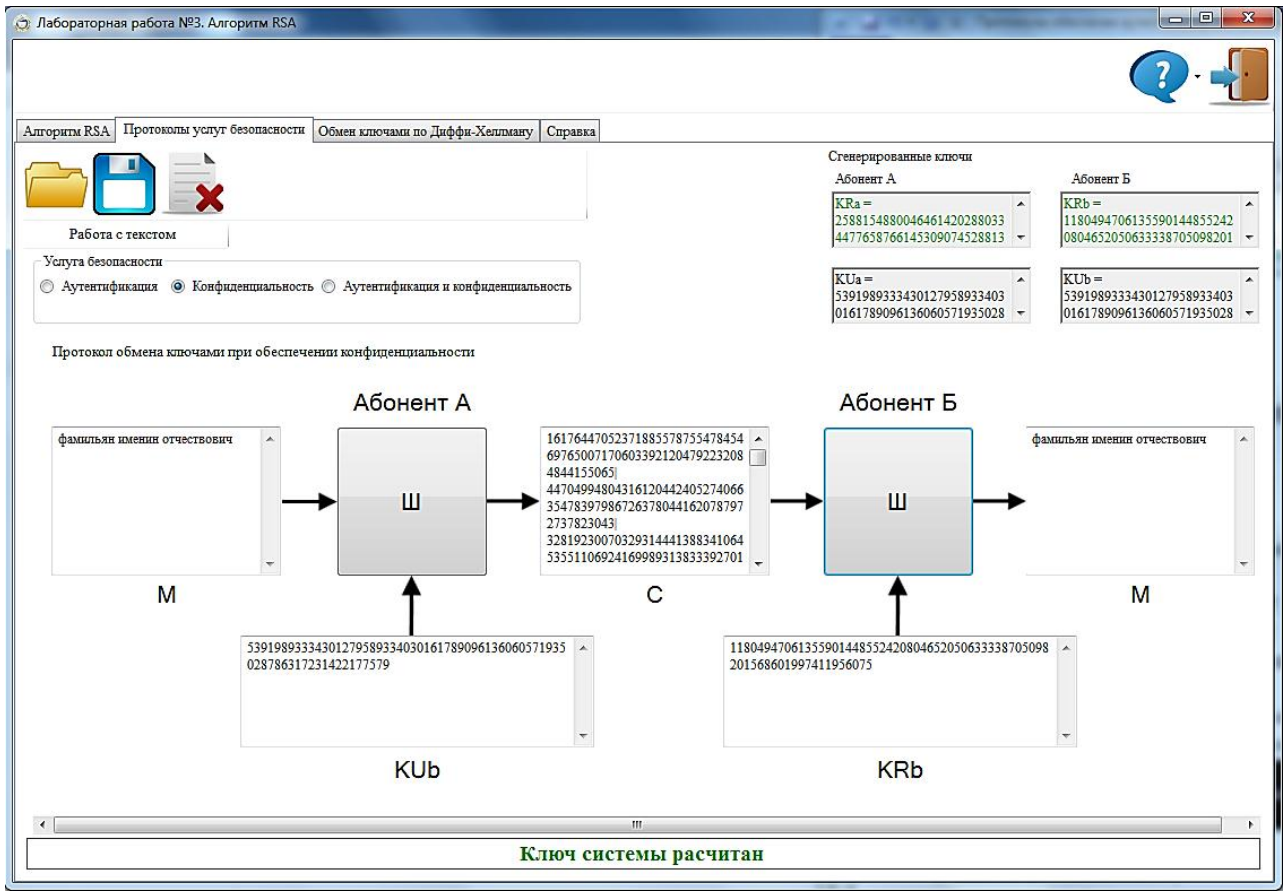


Рис. 3.11. **Розшифрування шифротексту**

Забезпечення автентичності та конфіденційності:

1. Виберіть режим "Аутентичность и конфиденциальность".
2. Уведіть вихідний текст.
3. Натисніть кнопку "Ш" (особистий ключ KR_A).
4. Натисніть кнопку "Ш" (особистий ключ KR_B).
5. Натисніть кнопку "Ш" (відкритий ключ KU_A).
6. Натисніть кнопку "Ш" (особистий ключ KU_B).
7. Розшифрований шифротекст має збігатися з вихідним текстом (рис. 3.12).

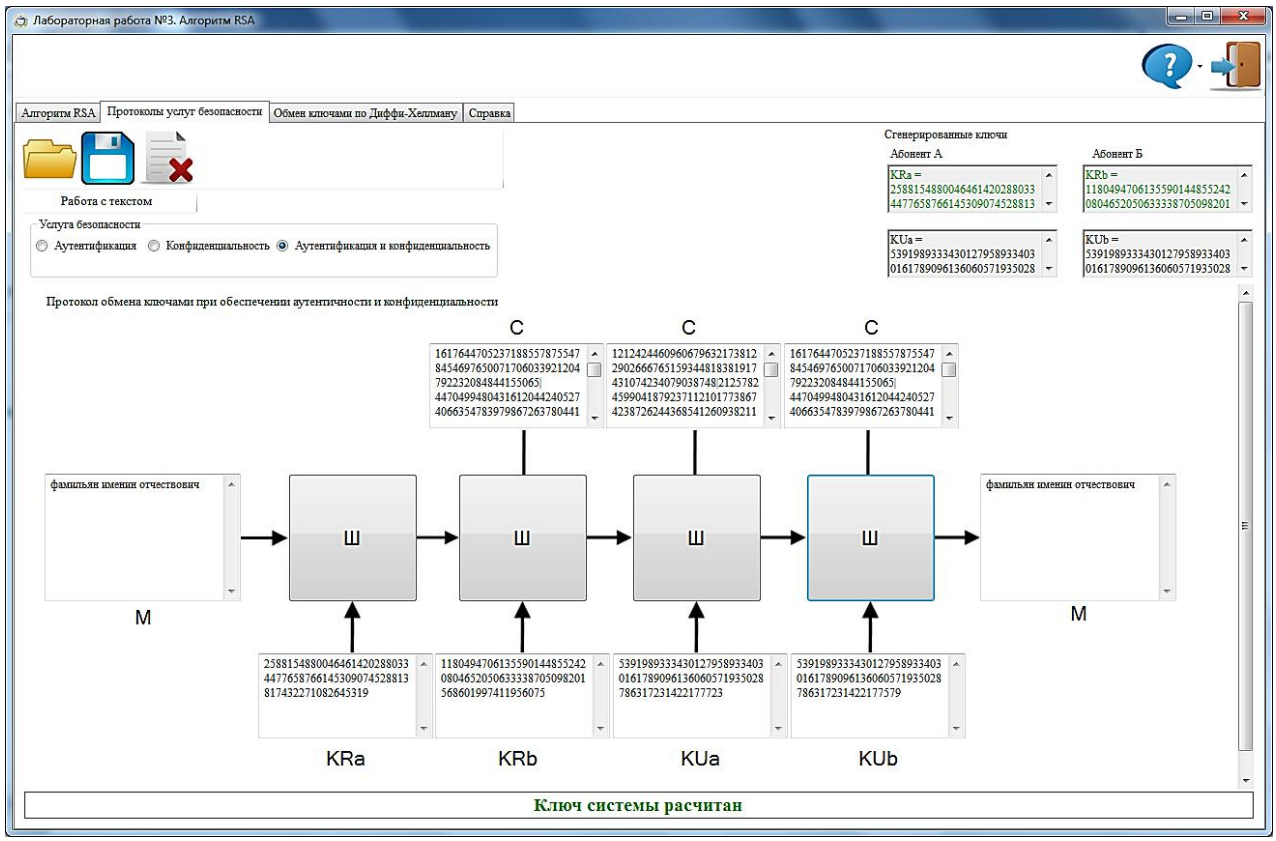


Рис. 3.12. **Розшифрування шифротексту**

3.5. Завдання до лабораторної роботи

Використовуючи програму, зашифруйте та розшифруйте повідомлення, яке містить ПІБ студента, слід користуватися таблицею 3.2.

Таблиця 3.2

Варіанти індивідуального завдання

Варіанти	Порядок простого числа q (біт)	Порядок простого числа p (біт)
1	2	3
1	896	768
2	864	832
3	832	896
4	800	960
5	768	1 024
6	736	1 088

Закінчення табл. 3.2

1	2	3
7	704	1 152
8	672	1 216
9	640	1 280
10	608	1 344
11	576	1 408
12	544	1 472
13	512	1 536
14	480	1 600
15	448	1 664
16	416	1 728
17	384	1 792
18	352	1 856
19	320	1 920
20	288	1 984

3.6. Контрольні запитання

1. Що таке "публічний ключ"?
2. Що таке "приватний ключ"?
3. Визначте основні поняття асиметричного шифрування.
4. Поясніть порядок формування параметрів в алгоритмі RSA.
5. Чим відрізняється асиметричне шифрування від симетричного?
6. Як генерують ключі в алгоритмі RSA?
7. Наведіть приклади односторонніх функцій.
8. У чому полягає сутність використання функцій із секретом.
9. Яким чином виконують додавання точок на еліптичній кривій.
10. Перелічіть параметри еліптичних кривих, що використовують у криптографії.
11. Який порядок формування ключів для асиметричного шифрування на еліптичних кривих?
12. Визначте порядок шифрування за алгоритмом ECIES.
13. Визначте порядок дешифрування за алгоритмом ECIES.
14. У чому полягає сутність функції KDF (key derivation function).

Лабораторна робота 4
Алгоритм цифрового підпису

4.1. Мета

Мета цієї лабораторної роботи – закріпити теоретичні знання та набути навичок у застосуванні та дослідженні систем цифрового підпису з використанням несиметричних криптоперетворень.

4.2. Рекомендації до підготовки до виконання

Для успішного виконання лабораторної роботи необхідно вивчити основні поняття та визначення з питань автентифікації; вивчити систему автентифікації на основі цифрового підпису Ель-Гамаля; ознайомитися з описом лабораторної роботи та програмним забезпеченням; підготувати відповіді на контрольні запитання.

4.3. Загальні теоретичні положення

Цифровий підпис (ЦП) повинен мати такі самі властивості, що й чорнильний. На відміну від асиметричного шифрування, ЦП реалізує такі послуги безпеки, як цілісність та автентичність.

Цифровий підпис – це цифровий еквівалент підпису, наявність якого в повідомленні дозволяє з високою вірогідністю визначити автора цього повідомлення.

Цифровий підпис є зашифрованою сукупністю даних, що містять:

- дані відправника;
- дані отримувача;
- час створення підпису;
- контрольну суму файлу;
- підпис відправника.

Стандарт цифрового підпису DSA

Алгоритм DSA було взято як федеральний стандарт цифрового підпису у США 1991 року. Після кількох модифікацій його було остаточно затверджено стандартом для несекретних застосувань 1994 року.

Алгоритм *DSA* (Digital Signature Algorithm) є варіантом цифрового підпису Шнорра – Ель-Гамаля. Він використовує такі параметри:

- p – просте число довжиною L бітів, де L може набувати значень від 512 до 1 024 (у першому варіанті алгоритму p мало фіксоване значення 512 бітів, що критикували криптографи як мале значення модуля). Значення p має бути кратним 64;
- число q, що є дільником p – 1 (щонайменше 160 бітів у двійковому поданні);
- елемент h із множини $\{1, p\}$ порядку q;
- випадкове число $a < q$;
- число $b = h^a \bmod p$.

Алгоритм також використовує геш-функцію $H(M)$. Стандарт потребує використання SHA.

Величини p, q, h і b є відкритими й утворюють публічний ключ.

Приватним ключем у такому разі буде число a.

Для формування електронного підпису повідомлення M власник приватного ключа виконує такі дії:

а) вибирає випадкове число $r < q$;

б) обчислює $r` = r^{-1} \bmod q$;

в) обчислює $s_1 = (h^r \bmod p) \bmod q$;

г) обчислює $s_2 = r`(H(M) + as_1) \bmod q$;

д) надає пару чисел (s_1, s_2) як підпис для повідомлення M.

Перевірку електронного підпису зведено до обчислення отримувачем:

а) $s` = s_2^{-1} \bmod q$;

б) $u_1 = H(M)s` \bmod q$;

в) $u_2 = s`s_1 \bmod q$;

г) $t = (h^{U1}b^{U2} \bmod p) \bmod q$;

У разі, якщо $t = s_1$ – підпис є справжнім.

Доведення коректності процедури підписування

Коректність процедури ґрунтується на тому, що $h^{U1}b^{U2} \bmod p$.

Справді, оскільки $b = h^a \bmod p$, то $h^r = h^{U1}b^{U2} \bmod p$; $h^r = h^{U1+aU2} \bmod p$. Отже, для доведення треба показати, що $r = (u_1 + au_2) \bmod q$. Це рівнозначне доведенню істинності співвідношення $1 \equiv r`(u_1 + au_2) \bmod q$.

Підставляннням сюди значення u_1u_2, визначено:

$$r`(u_1 + au_2) \equiv r`(H(M)s` + as`s_1) \bmod q \equiv r`(H(M) + as_1)\, s` \bmod q \equiv s_2s` \bmod q \equiv$$
$$\equiv s_2s_2^{-1} \bmod q \equiv 1.$$

Отже, процедури утворення та перевірки підпису коректні.

Безпека DSA. Як уже згадували, основним недоліком першої версії DSA була замала довжина модуля *p*, усього 512 бітів. Тому в подальших редакціях цього алгоритму було вирішено надати можливість користувачам змінювати довжину модуля від 512 до 1 024 бітів, що значно підсилило криптостійкість алгоритму до атаки "грубою силою". Отже, алгоритм цифрового підпису DSA можна вважати стійкішим за решту алгоритмів, що утворюють 128-бітний геш-образ (рис. 4.1).

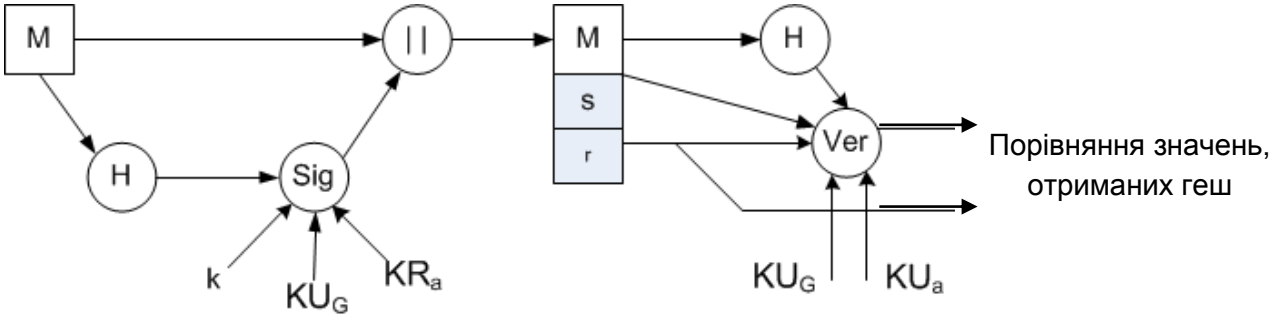


Рис. 4.1. **Цифровий підпис стандарт DSA**

Публічний ключ системи: *p*, *q* – прості числа *q p*, $1\ q < 2^{512}$, $1\ p < 2^{1024}$.

Відправник виконує такі дії:

1) генерує особистий ключ – випадкове ціле число (KR_A) x, $1 < X < (p - 1)$;

2) обчислює публічний ключ $y = q^x \bmod P$ (KU_A);

3) обчислює геш-код повідомлення $m = h(M)$, $1 < m < (p - 1)$;

4) генерує випадкове ціле число K, $1 < K < (p - 1)$ так, щоб $gcd(K, (p - 1)) = 1$;

5) обчислює $r = q^K \bmod P$;

6) обчислює $s = (km + xr) \bmod (p - 1)$. Пара чисел (r, s) утворює цифровий підпис $S = (r, s)$;

7) у канал зв'язку (M, r, s).

Абонент-отримувач:

1) обчислює за прийнятим повідомленням M число $m' = h(M)$;

2) обчислює значення $A = y * r^s \pmod p$;

3) обчислює значення $A' = q^{m'} \pmod p$;

4) порівнює дані значення, перевіряє справедливість співвідношення $y * r^s \pmod p = q^{m'} \pmod p$.

Стандарт цифрового підпису DSA

Національний інститут стандартів і технології США (NIST) розробив федеральний стандарт цифрового підпису DSS. Для створення цифрового підпису використовується алгоритм DSA (Digital Signature Algorithm). Як геш-алгоритм стандарт передбачає використання алгоритму SHA-1 (Secure Hash Algorithm). DSS спочатку було запропоновано 1991 року та переглянуто 1993 року у відповідь на публікації, що стосуються безпеки його схеми.

Підхід DSS. DSS використовує алгоритм, який розроблявся для використання тільки як цифровий підпис (рис. 4.2). На відміну від RSA, його не можна використовувати для шифрування або обміну ключами. Тим не менш, це технологія відкритого ключа.

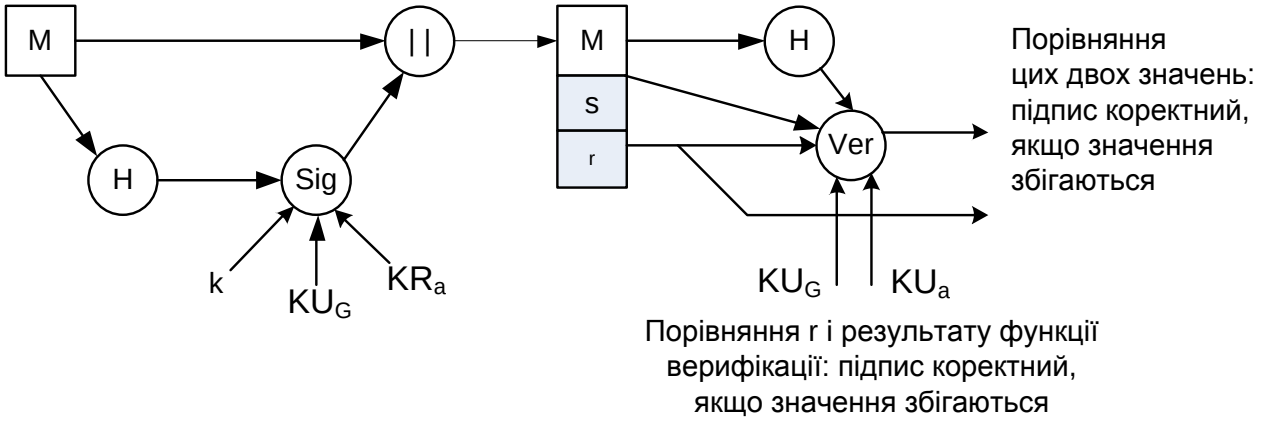


Рис. 4.2. **Створення та перевірка підпису за допомогою стандарту DSS**

Підхід DSS використовує сильну геш-функцію. Геш-код є входом функції підпису разом із випадковим числом *k*, створеним для цього конкретного підпису. Функція підпису також залежить від приватного ключа відправника KR_a та багатьох параметрів, відомих усім учасникам. Можна вважати, що вони утворюють глобальний відкритий ключ KU_G. Результатом є підпис, що складається із двох компонент – *s* і *r*.

Для перевірки підпису отримувач також створює геш-код отриманого повідомлення. Цей геш-код разом із підписом подають на вхід функції верифікації. Вона залежить від глобального відкритого ключа KU_G і від публічного ключа відправника KU_a. Виходом функції верифікації є значення, яке має дорівнювати компоненті підпису *r*, якщо підпис коректний.

Функція підпису така, що тільки відправник, який знає приватний ключ, може створити коректний підпис.

Тепер необхідно розглянути деталі алгоритму DSS. Він ґрунтується на складності обчислення дискретних логарифмів.

Загальні параметри групи користувачів. Алгоритм має три відкритих параметри, загальних для великої групи користувачів:

- 160-бітове просте число q, тобто $2^{159} < q < 2^{160}$;
- просте число p довжиною між 512 і 1 024 бітами має бути таким, щоб q було дільником $(p - 1)$, тобто $2^{L-1} < p < 2^{L}$, де $512 < L < 1\,024$ і $(p - 1) / q$ є цілим;
- $g = h^{(p-1)/q} \bmod p$, де h є цілим між 1 і $(p - 1)$.

Знаючи ці числа, кожен користувач вибирає приватний ключ і створює публічний ключ.

Приватний ключ відправника. Приватний ключ x має бути числом між 1 і $(q - 1)$ та бути випадковим або псевдовипадковим числом:

x – випадкове або псевдовипадкове ціле, $0 < x < q$.

Публічний ключ відправника. Публічний ключ обчислюють із приватного ключа як $y = g^x \bmod p$. Обчислити y за відомим x досить просто. Однак, маючи публічний ключ y, обчислюванням неможливо визначити x, який є дискретним логарифмом y по основі g: $y = g^x \bmod p$.

Випадкове число, унікальне для кожного підпису. Алгоритм використовує k-випадкове або псевдовипадкове ціле $0 < k < q$, унікальне для кожного підписування.

Підписування. Для створення підпису відправник обчислює дві величини, r і s, які є функцією від компонентів публічного ключа (p, q, g), приватного ключа користувача (x), геш-коду повідомлення $H(M)$ і k:

$$r = (g^k \bmod p) \bmod q;$$
$$s = [k^{-1}(H(M) + xr)] \bmod q \text{ - } \textit{підпис} = (r, s).$$

Перевірка підпису. Отримувач виконує перевірку підпису з використанням певних формул. Він створює величину v, яка є функцією від компонентів загального відкритого ключа, публічного ключа відправника та геш-коду отриманого повідомлення. Якщо ця величина дорівнює компоненту r у підписі, то підпис уважають дійсним:

$$W = s^{-1} \bmod q;$$

$$u1 = [H(M)w] \bmod q^{u2} = rw \bmod q;$$
$$v = [(g^{u1}y^{u2}) \bmod p] \bmod q.$$

Підпис є коректним, якщо $v = r$.

Алгоритм обчислення

Публічний ключ системи:

- p, q – прості числа, $2^{159} < q < 2^{512}$, $2^{512} < p < 2^{1024}$;

q дільник $(p - 1)$; h – просте число, $1\ h < (p - 1)$, $g = h(p - 1) / q \bmod p$.

Як геш-функції використовують алгоритм SHA-1 (SHA-256).

Відправник:

1) генерує приватний ключ – випадкове ціле число $(KR_A) \Longrightarrow x$, $0 < x < q$;

2) обчислює публічний ключ $(KU_A) \Longrightarrow y = g^x \bmod p$;

3) обчислює геш-код повідомлення $m = h(M)$, $1 < m < (p - 1)$;

4) генерує випадкове ціле число K, $0 < K < q$ так, щоб $gcd(K, q) = 1$;

5) обчислює $r = (g^k \bmod p) \bmod q$;

6) обчислює $s = [k^{-1} m + xr] \bmod q$. Пара чисел (r, s) утворює цифровий підпис $S = (r, s)$;

7) у канал зв'язку відправляють величини (M, r, s).

Отримувач виконує такі дії:

1) обчислює за прийнятим повідомленням M число $m = h(M)$;

2) обчислює значення: $w = s^{-1} \bmod q$;

$$u1 = [m/w] \bmod q;$$
$$u2 = [\ r\ w\ ] \bmod q;$$

3) обчислює значення $v = [(g_{u1}\ y_{u2}) \bmod p] \bmod q$;

4) підпис є коректним, якщо $v = r \Leftrightarrow$ порівняння значень в операції *XOR*.

Цифровий підпис ДСТУ 4145-2002

Алгоритм електронного цифрового підпису стандарту ДСТУ 4145-2002 ґрунтується на еліптичних кривих над полем Галуа (несуперсингулярні криві). У стандарті використовують криві в поліноміальному базисі та криві в оптимальному нормальному базисі, причому останні (ОНБ) – другого та третього типу. Основою для розроблення цього стандарту був міжнародний стандарт IEEEP 1363a. Тому принциповою відмінністю ДСТУ 4145-2002 є тільки використання "своєї" функції гешування. Для гешування використовують алгоритм – Міждержавний стандарт ГОСТ 34.311-95 (раніше ГОСТ 28147-89 у режимі імітовставки).

Цей стандарт установлює механізм цифрового підпису на властивостях груп точок еліптичних кривих над полями $GF(2^m)$ і правила застосування цього механізму до повідомлень, які передають каналами зв'язку й/або обробляють у комп'ютеризованих системах загального призначення. Застосування цього стандарту гарантує цілісність підписаного повідомлення, автентичність його автора та незаперечність авторства за умови дотримання певних правил використання цифрового підпису, які не є об'єктом цього стандарту. Далі визначено правила реалізації процедур необхідних для побудови криптографічних алгоритмів, визначених цим стандартом.

Генератор псевдовипадкових послідовностей. Генератор псевдовипадкових послідовностей використовують для визначення випадкових даних, необхідних для побудови загальних параметрів цифрового підпису, обчислення приватних і публічних ключів цифрового підпису й обчислення самого цифрового підпису.

Такий генератор рекомендовано використовувати уповноваженим органом державного управління.

Функція гешування. У цьому стандарті функцію гешування використовують в обчисленні й перевірці цифрового підпису. Функція гешування H перетворить текст T довжини L_m у двійковий рядок $H(T)$ фіксованої довжини L_H.

У цьому стандарті мають використовувати функцію гешування, визначену ГОСТ 34.311-95, або будь-яка інша, рекомендована уповноваженим органом державного управління. Значення параметра L_H однозначно визначено ідентифікатором i_h конкретної функції, яку використовують разом із цим стандартом. Параметр $\boldsymbol{i}_h$ входить до загальних параметрів цифрового підпису. Таких параметрів у групі користувачів може бути кілька; водночас $L_H \geq 160$, $L(I_H) \leq 64$. Значення $I_H = 1$, $L(I_H) = 8$ відповідають функції гешування, визначеній ГОСТ 34.311-95.

Дозволено використання геш-функції за замовчуванням. У цьому разі параметра L_m може не бути. Якщо використана функція накладає обмеження на довжину тексту L_m, то ці обмеження мають силу й для цього стандарту.

Обчислення випадкового цілого числа. Необхідно визначити алгоритм обчислення випадкового цілого числа a, щоб $L(a) < L(n)$. Вихідні дані алгоритму: порядок базової точки еліптичної кривої n; довжина t послідовності, яка створюється датчиком псевдовипадкових послідовностей

за одним звертанням до нього. Результат виконання алгоритму – випадкове ціле число a, що задовільнює умову $L(a) < L(n)$.

Алгоритм обчислення випадкового цілого числа:

1) обчислюють довжину $L(n)$ двійкового подання цілого числа n;

2) обчислюють мінімальне значення k, щоб $k_t \geq L(n) - 1$;

3) за k звертань до генератора псевдовипадкових послідовностей формують випадковий двійковий рядок довжини k_t; перші $l(n) - 1$ елементів цієї послідовності утворюють випадковий двійковий рядок $R_{L(n)2}, \ldots, R_0$ довжиною $l(n) - 1$;

4) уважають, що $\alpha_i = R_i$ для $i = 0, \ldots, l(n) - 2$;

5) визначають індекс j, який дорівнює найбільшому значенню індексу i, для якого $a_i = 1$; якщо такого індексу немає, то вважають, що $j = 0$, $a_0 = 0$;

6) випадковий рядок $R_{L(n)2}, \ldots, R_0$ знищують;

7) двійковий рядок $\alpha_j, \ldots, \alpha_0$ задає випадкове ціле число a довжиною $j + 1$.

Обчислення випадкового елемента базового поля. Необхідно визначити алгоритм обчислення випадкового елемента x базового поля $GF(2^m)$. Вихідні дані алгоритму: степінь базового поля m; довжина t псевдовипадкової послідовності, яка створюється генератором псевдовипадкових послідовностей за одним звертанням до нього.

Результат виконання алгоритму – випадковий елемент базового поля m.

Алгоритм обчислення випадкового елемента базового поля:

1) обчислюють мінімальне значення k, щоб $k_t \geq m - 1$;

2) за k звертань до датчика псевдовипадкових послідовностей формують випадковий двійковий рядок довжини k_t;

3) перші m елементів цієї послідовності утворюють випадкову двійкову послідовність $R_{m-1}, \ldots R_0$;

4) уважають, що $x_i = R_i$ для $i = 0, \ldots, m - 1$;

5) випадковий рядок $R_{m-1}, \ldots R_0$ знищується;

6) двійковий рядок $x_{m-1}, \ldots x_0$ задає випадковий елемент базового поля x.

Обчислення сліду елемента базового поля. Вихідні дані алгоритму: елемент x базового поля $GF(2^m)$.

Результат виконання алгоритму – слід $tr(x)$ елемента x.

Алгоритм обчислення сліду:

1) уважають, що $t = x$;

2) для i від 1 до $m - 1$ обчислюють $t \leftarrow t^2 + x$;

3) результат обчислення сліду $tr(x) = t$.

Обчислення напівсліду елемента базового поля. Необхідно визначити алгоритм обчислення напівсліду елемента x базового поля, непарного ступеня m. Результат виконання алгоритму – напівслід $htr(x)$ елемента x.

Алгоритм обчислення напівсліду:

1) уважають, що $t = x$;

2) для i від 1 до $(m - 1) / 2$ обчислюють $t \leftarrow t^4 + x$;

3) результат обчислення напівсліду $htr(x) = t$.

Розв'язання квадратного рівняння в базовому полі. Вихідні дані алгоритму: квадратне рівняння $z^2 + uz = w$, $u,w \in GF(m^2)$. Результат виконання алгоритму – кількість розв'язків k квадратного рівняння й один із розв'язків цього рівняння, якщо $k > 0$.

Алгоритм розв'язання квадратного рівняння:

1) якщо $u = 0$, то $z = w^{2m-1} = \sqrt{w}, k = 1$ і виконують перехід до кроку 8;

2) якщо $w = 0$, то вважають, що $k = 2$, $z = 0$ і виконують перехід до кроку 8;

3) обчислюють елемент базового поля $v = wu^{-2}$;

4) обчислюють слід елемента v;

5) якщо слід елемента $tr(v) = 1$, то вважають, що $k = 2$, $z = 0$ і виконують перехід до кроку 8;

6) обчислюють напівслід $t = htr(v)$ елемента v;

7) обчислюють елемент базового поля $z = tu$ і вважають, що $k = 2$;

8) результат виконання алгоритму: кількість розв'язків k квадратного рівняння та розв'язок цього рівняння z, якщо $k > 0$.

Обчислення випадкової точки еліптичної кривої. Вихідні дані алгоритму: еліптична крива $y^2 + xy = x^3 + Ax^2 + B$ над полем $GF(m^2)$, $B \neq 0$, $A \in \{0, 1\}$.

Результат виконання алгоритму – випадкова точка цієї еліптичної кривої $P = (x_p, y_p)$.

Алгоритм обчислення випадкової точки еліптичної кривої:

1) обчислюють випадковий елемент i базового поля;

2) обчислюють елемент базового поля $W = U^3 + Au^2 + B$;

3) розв'язують квадратне рівняння $z^2 + uz = w$;

4) якщо число розв'язків квадратного рівняння дорівнює 0, то виконують перехід до кроку 1; якщо ні, то виконують перехід до кроку 5;

5) уважають, що $x_p = u$, $y^p = z$, z – розв'язок квадратного рівняння, визначений на кроці 3;

6) результат виконання алгоритму – випадкова точка еліптичної кривої p із координатами (x_p, y_p).

Стиск точки еліптичної кривої. Необхідно визначити алгоритм перетворення точки P простого порядку n еліптичної кривої з координатами (x_p, y_p) у стисле подання $P \in GF(m^2)$, m – непарне число. Вихідні дані алгоритму: точка еліптичної кривої P простого порядку n із координатами (x_p, y_p).

Результат виконання алгоритму – стисле подання $P \in GF(m^2)$ точки еліптичної кривої.

Алгоритм стиску точки еліптичної кривої:

1) якщо $x_p = 0$, то вважають, що $P = 0$ і виконують перехід до кроку 3;

2) якщо $x_p \neq 0$, то обчислюють елемент базового поля $y = y_p x_p^{-1} = \left(y_{m-1}, \ldots, y_0\right)$, обчислюють слід елемента y, $i = tr(y)$ і вважають, що $P = \left(P_{m-1}, \ldots, P_0\right) = \left(X_{p,m-1}, \ldots, X_{p,1}, i\right)$, тобто крайній правий двійковий розряд координати X_p заміняють значенням сліду елемента y;

3) результат виконання алгоритму: $P \in GF(m^2)$ – стисле подання точки p еліптичної кривої.

Обчислення ключів цифрового підпису. Необхідно встановити порядок обчислення приватного d і публічного Q ключів цифрового підпису.

Обчислення приватного ключа цифрового підпису. Приватний ключ d цифрового підпису обчислюють у такий спосіб:

1) обчислюють випадкове ціле число d;

2) якщо $d \neq 0$, то його беруть за приватний ключ цифрового підпису, а якщо ні, то генерують наступне число.

Умови обчислення та зберігання приватного ключа цифрового підпису мають виключати можливість несанкціонованого доступу до нього або його частини, а також до даних, які використовували у процесі його обчислення. Умови зберігання приватного ключа мають виключати можливість його модифікації, знищення або підміни.

Обчислення публічного ключа цифрового підпису. Публічний ключ цифрового підпису є точкою еліптичної кривої вигляду $Q = -dP$, де P – базова точка еліптичної кривої, d – приватний ключ цифрового підпису.

Умови зберігання публічного ключа мають виключати можливість його модифікації або підміни. Допускають зберігання та передавання публічного ключа цифрового підпису у стислому вигляді.

Перевірка коректності ключів цифрового підпису. Високу криптографічну стійкість цифрового підпису, обчисленого, згідно із цим стандартом, гарантовано тільки в тому разі, якщо приватний та публічний ключі цифрового підпису обчислені коректно, тобто точно, відповідно до цього стандарту.

Перевірка коректності відкритого ключа цифрового підпису. Публічний ключ Q має задовольняти такі умови:

1) координати точки еліптичної кривої, що є відкритим ключем цифрового підпису, належать до базового поля, тобто є двійковими числами довжини m;

2) $Q \neq 0$;

3) публічний ключ $Q = (x_Q, y_Q)$ лежить на еліптичній кривій, тобто його координати задовольняють рівняння еліптичної кривої;

4) порядок публічного ключа $Q = (x_Q, y_Q)$ дорівнює n, тобто $nQ = 0$.

Якщо умови 1 – 4 виконано, то публічний ключ цифрового підпису є коректним.

Зазначену перевірку можна не виконувати, якщо використані в конкретній реалізації цифрового підпису способи зберігання публічного ключа цифрового підпису виключають можливість його підміни, модифікації або знищення.

Перевірку коректності приватного ключа виконує виключно його власник у такий спосіб:

1) обчислює точку еліптичної кривої $Q' = -dP$, де P – базова точка еліптичної кривої, d – приватний ключ ЕЦП;

2) якщо $Q' = Q$, де Q – публічний ключ цифрового підпису. Тоді приватний ключ відповідає вибраному публічному ключу цифрового підпису.

Таку перевірку можна не виконувати, якщо використані способи зберігання приватного ключа виключають можливість його підміни, модифікації або знищення.

Обчислення цифрового передпідпису. Вихідні дані алгоритму: загальні параметри цифрового підпису. Результат виконання алгоритму – цифровий передпідпис (e, F_e), $e \in GF(n)$ – ціле число, $F_e \in GF(m^2)$.

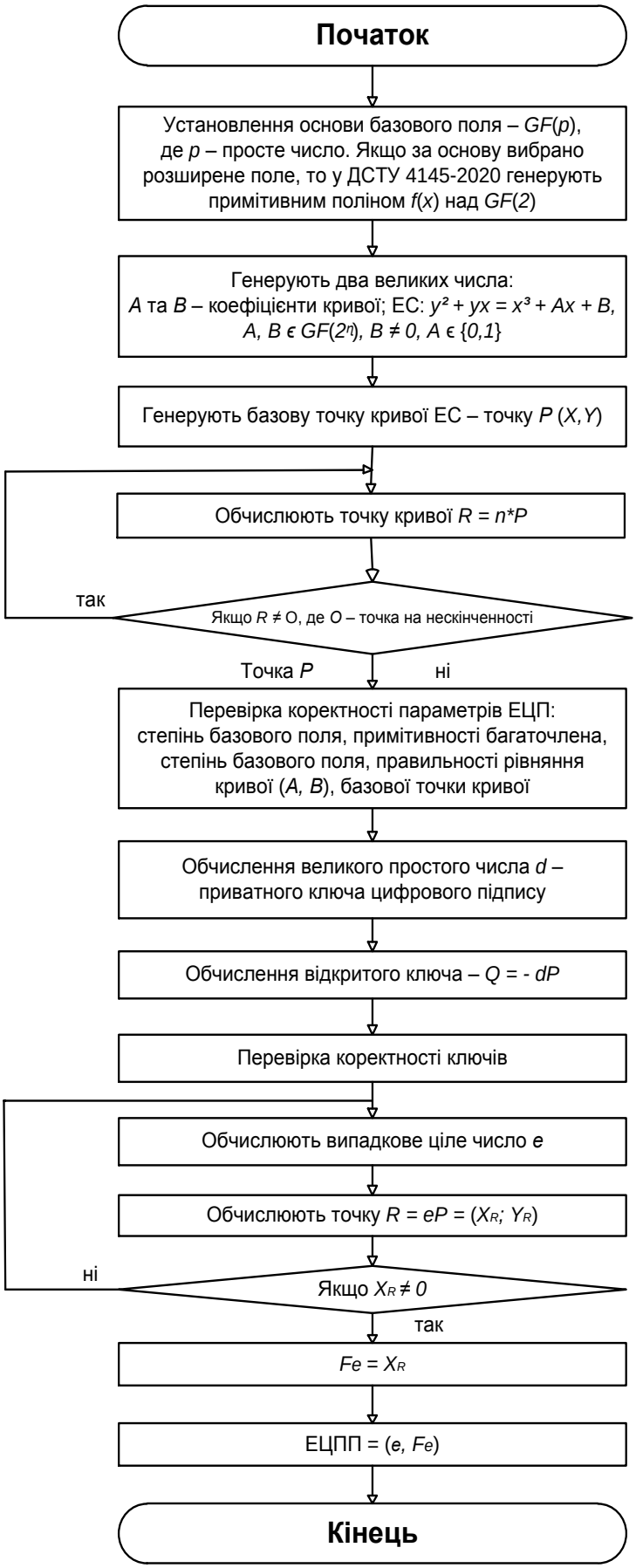
Алгоритм обчислення цифрового передпідпису:

1) обчислюють випадкове ціле число e;
2) обчислюють точка еліптичної кривої $R = eP = (x_R, y_R)$;
3) якщо координата $xR = 0$, то виконують перехід до кроку 1; якщо ні, то вважають, що $F_e = xR$, і виконують перехід до кроку 4;
4) результат виконання алгоритму – цифровий передпідпис (e, F_e).

На рис. 4.3 та 4.4 наведено алгоритм електронного цифрового підпису стандарту ДСТУ 4145-2002.

Допускають попереднє обчислення довільного числа цифрових передпідписів. Після використання цифрового предпідпису у процедурі цифрового підпису він негайно знищується.

Цифровий передпідпис секретний. Умови його обчислення та зберігання мають виключати можливість несанкціонованого доступу до нього або його частини, а також до даних, які використовували у процесі обчислення. Умови зберігання цифрового передпідпису мають виключати можливість його модифікації або підміни.



Незведений поліном – це поліном, який ділиться тільки на себе та на одиницю. Примітивним поліномом називають поліном f(x), корінь якого є примітивним елементом поля.

Рис. 4.3. **Частина алгоритму електронного цифрового підпису**

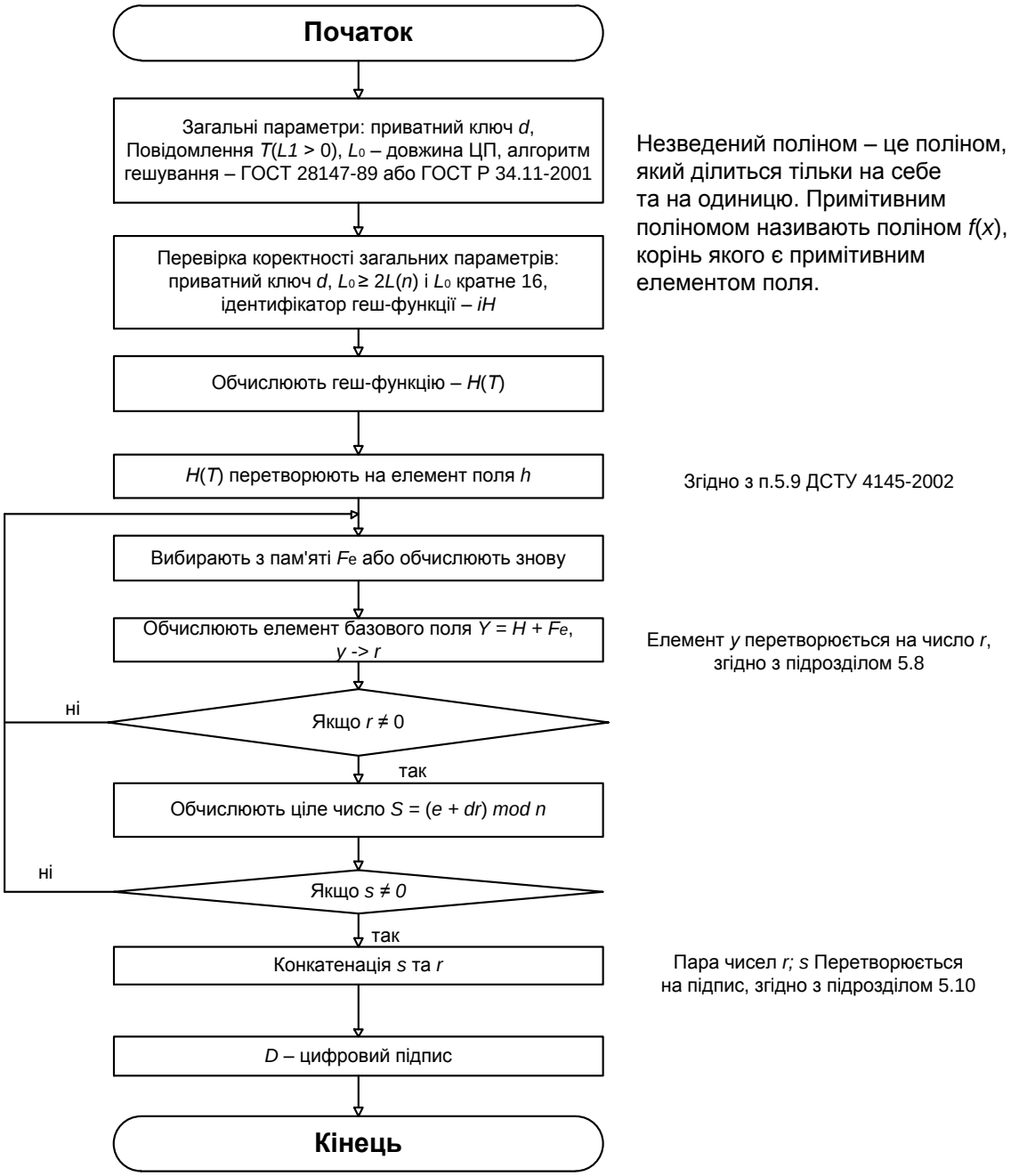


Рис. 4.4. **Частина алгоритму електронного цифрового підпису**

Стандарти ЕЦП ГОСТ Р 34.10-94 та ГОСТ Р 34.10-2001

Алгоритм ГОСТ Р 34.10-94 дуже схожий на DSA та використовує такі параметри:

p – просте число, довжина якого може бути між 509 і 512 бітами або 1 020 – 1 024 бітів;

q – просте число, множник *p* - *1*, довжиною 254 – 256 бітів;

a – будь-яке число, менше за *p* - *1*, для якого $a^q \bmod p = 1$;

x – довільне число, менше за *q*;

$y = a^x \bmod p$.

Перші три параметри – p, q, a – відкриті та можуть використовуватися всіма користувачами мережі. Величина y – публічний ключ, а x – приватний.

Щоб підписати повідомлення M, необхідно виконати такі кроки:

1) відправник генерує випадкове число k, менше за q;

2) відправник обчислює два числа: $r' = a^k \bmod p$ і $r = r' \bmod q$. Якщо $r = 0$, то генерують інше число k;

3) із використанням приватного ключа користувача обчислюють: $s = (xr + kH(M)) \bmod q$. Якщо $s = 0$, то генерують нове число k;

4) каналом зв'язку надсилають повідомлення M разом із підписом s.

Рівняння перевірки підпису в такому разі буде таким:

$$r = \left(a^{sH(M)^{-1}} y^{-rH(M)^{-1}} \bmod p\right) \bmod q. \qquad (4.1)$$

Слід зазначити, що повідомлення, яке дає нульове значення геш-образу $H(M)$, не підписують, оскільки в такому разі рівняння перевірки підпису спрощується настільки, що зловмисник може легко обчислити приватний ключ.

Із підвищенням продуктивності процесорів і розробленням нових типів криптоаналітичних атак виникла потреба у зміцненні алгоритму цифрового підпису. Тому стандарт ГОСТ Р 34.10-94 було модифіковано. Новий стандарт, а саме ГОСТ Р 34.10-2001, було впроваджено в дію 1 липня 2002 року замість попереднього.

У цьому стандарті використовують алгоритм з операціями над кінцевим полем групи точок еліптичної кривої. Використовують еліптичну криву E у формі Вейєрштраса над простим полем, криву задають коефіцієнтами a та b або її інваріантом:

$$J(E) = 1728 \frac{4a^3}{4a^3 + 27b^2} \bmod p. \qquad (4.2)$$

Коефіцієнти a та b визначають за відомим інваріантом таким чином:

$$a \equiv 3k \bmod p;$$
$$b \equiv 2k \bmod p, \qquad (4.3)$$

де $k = \dfrac{J(E)}{1728 - J(E)}$; $J(E) \neq 0$; $1\,728$.

Точку Q слід називати точкою кратності k, якщо для деякої точки P справджується рівність $Q = kP$.

Параметрами такої схеми ЕЦП будуть такі значення:

1) p – модуль еліптичної кривої, просте число $p > 2^{255}$;

2) еліптична крива, задана інваріантом $J(E)$ або коефіцієнтами a та b;

3) ціле число m – порядок групи точок еліптичної кривої E;

4) просте число q – порядок циклічної підгрупи групи точок еліптичної кривої E, для якого виконують такі умови:

- $m = nq,\ n \in Z,\ n \geq 1;\ 2^{254} < q < 2^{256}$;
- базова точка $P \neq O$ на кривій порядку q (тобто $qP = O$). Координати цієї точки позначте через (x_p, y_p);
- геш-функція, передбачена стандартом ГОСТ Р 34.11-94, перетворює двійкову послідовність довільної довжини у двійкову послідовність довжиною 256 бітів (у цьому стандарті як геш-функцію рекомендовано використання шифру ГОСТ 28147-89).

Кожен учасник інформаційного обміну повинен мати власну пару ключів:

- приватний ключ, ціле число $d,\ 0 < d < q$;
- публічний ключ, точка Q із координатами (x_p, y_p), що задовольняють рівняння $dP = Q$.

Для формування цифрового підпису необхідно виконати такі дії:

1) обчислити геш-образ повідомлення M, $h(M)$;

2) двійковому вектору $h = (\alpha_{255}, \ldots, \alpha_0)$ поставити у відповідність число:

$$\alpha = \sum_{i=0}^{255} \alpha_i 2^i; \tag{4.4}$$

3) згенерувати випадкове число $0 < k < q$;

4) обчислити точку еліптичної кривої $C = kP$ і визначити величину $r \equiv x_C \bmod q$, де x_C – x-координата точки C. Якщо $r = 0$, то згенерувати наступне випадкове число r ;

5) обчислити значення $s \equiv (rd + ke) \bmod q$. Якщо $s = 0$, тоді вибрати нове число k;

6) обчислити двійкові вектори, що відповідають числам r і s. Визначити цифровий підпис як конкатенацію цих двійкових подань:

$$\xi = \{r \,||\, s\}. \tag{4.5}$$

Для перевірки справжності ЕЦП виконують такі дії:

1) за знайденим значенням ЕЦП ξ відновлюють значення r і s. Якщо $0 < r < q$, $0 < s < q$, то переходять до наступного кроку. Якщо ні, то підпис є несправжнім;

2) обчислюють геш-образ отриманого повідомлення $h(M)$;

3) обчислюють число α, двійковим поданням якого є вектор h і число $e \equiv \alpha \bmod q$. Якщо $e = 0$, то встановити $e = 1$;

4) обчислюють $v \equiv e^{-1} \bmod q$, $z_1 \equiv sv \bmod q$, $z2 \equiv -rv \bmod q$;

5) обчислюють точку еліптичної кривої $C = z_1P + z_2Q$ і визначають $R \equiv x_C \bmod q$, де x_C – x-координата точки C;

6) якщо виконується рівність $R = r$, то підпис є справжнім. В іншому разі підпис не підтверджено.

Схожим чином працюють й інші системи електронного цифрового підпису, які ґрунтуються на еліптичних кривих (наприклад, ECDSA). Вони привертають увагу криптографів, оскільки дозволяють конструювати "елементи" та "правила об'єднання", які формують групи. Властивості цих груп відомі достатньо добре, щоб використати їх у криптографічних застосунках. Однак групи точок еліптичних кривих, на відміну від інших, не мають характерних властивостей, що полегшують криптоаналіз. Наприклад, їм не властиве поняття "гладкість".

За допомогою еліптичних кривих можна реалізувати багато криптографічних алгоритмів із відкритими ключами, наприклад, Діффі – Гелльмана, Ель-Гамаля та ін.

4.4. Практичне виконання

Стандарт цифрового підпису DSA:

1. Для початку необхідно створити пару ключів (публічний і приватний). Для цього потрібно ввести розрядність простих чисел p і q, на яких ґрунтується криптостійкість системи, та натиснути кнопку "Расчет p и q".

2. Потім необхідно наказати ввести число x, яке буде задовольняти умову $1 < x < (P - 1)$ для генерації відкритого ключа y, та натиснути кнопку "Расчет y".

3. Після цього слід:

- увести вихідний текст у відповідне поле;

• вибрати алгоритм гешування з наведених: MD5; SHA1; SHA256; SHA384; SHA512;

• натиснути кнопку "Хешировать сообщение".

4. Далі необхідно:

• увести ціле просте число *K*;

• розрахувати число *r* ;

• розрахувати число *s*.

Для того щоб згенерувати цифровий підпис, потрібно натиснути кнопку "Сгенерировать подпись". Із допомогою кнопки "Конкатенация" формують підписане повідомлення.

1. Із натисканням "Проверить хеш-сумму" відбувається перевірка геш-образів вихідного та переданого текстів.

2. Далі обчислюють значення *A* та відбувається перевірка повідомлення на справжність. У кінці натискають кнопку "Верификация" (рис. 4.5).

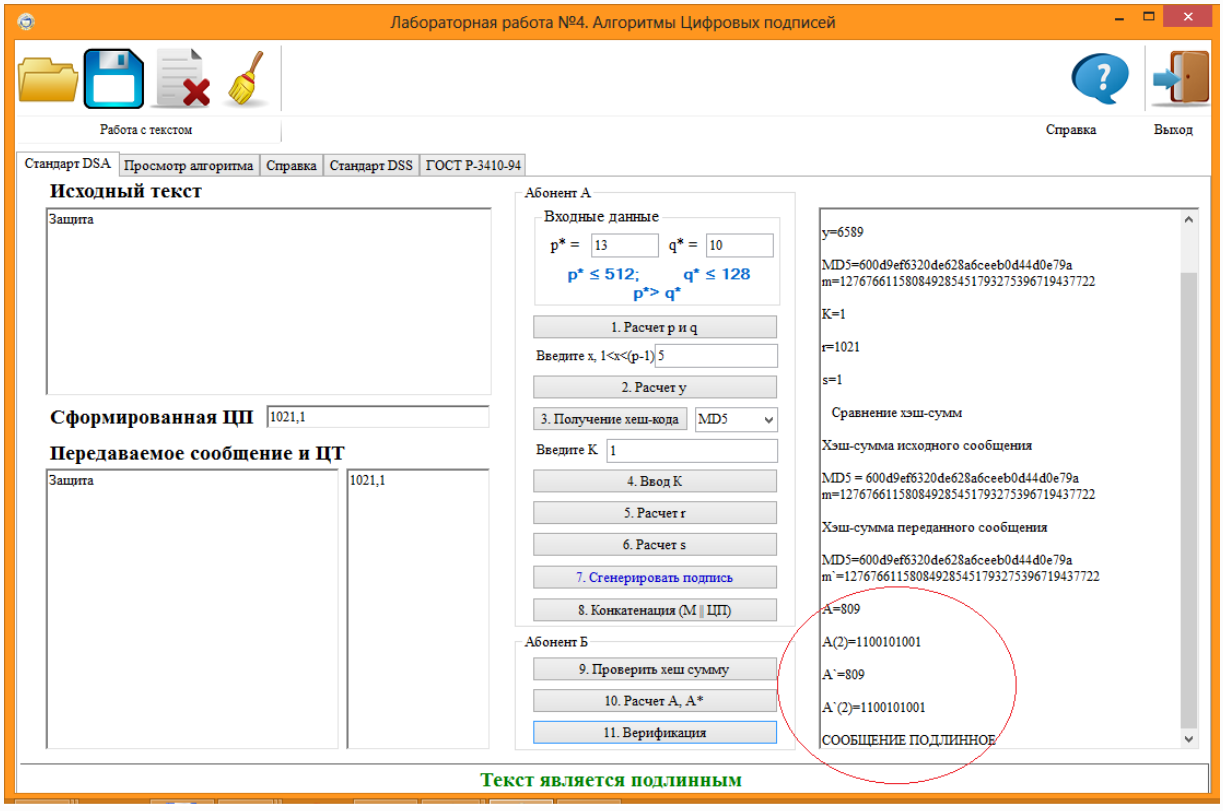


Рис. 4.5. **Формування цифрового підпису за стандартом DSA**

Стандарт цифрового підпису DSS:

1. Для початку необхідно створити пару ключів (публічний та приватний). Для цього потрібно ввести розрядність простих чисел *p*, *q* та *h*,

на яких ґрунтується криптостійкість системи. Далі натиснути кнопку "Расчет *p*, *q*, *g*".

2. Потім необхідно ввести число *x*, яке буде задовольняти умову *1 < x <* (*P - 1*) для генерування публічного ключа *y* та натиснути кнопку "Расчет *y*".

3. Після цього слід:

- увести вихідний текст у відповідне поле;
- вибрати алгоритм гешування з наведених: *SHA-1*; *SHA-256*;
- натиснути кнопку "Хешировать сообщение".

4. Далі необхідно:

- увести ціле просте число *K*;
- розрахувати число *r* ;
- розрахувати число *s*.

Для того щоб згенерувати цифровий підпис, необхідно натиснути кнопку "Сгенерировать подпись". Із допомогою кнопки "Конкатенация" формують підписане повідомлення.

5. Із натисканням "Проверить хеш-сумму" відбувається перевірка геш-образів вихідного та переданого текстів.

6. Далі обчислюють значення *A* та відбувається перевірка повідомлення на справжність. Для цього треба натиснути кнопку "Верификация".

Стандарти ЕЦП ГОСТ Р 34.10-94 і ГОСТ Р 34.10-2001:

1. Необхідно згенерувати пару ключів: публічний та приватний. Для цього потрібно ввести розрядність простих чисел *p*, *q і g*, на якій грунтується крипостійкість системи, та натиснути кнопку "Расчет *p* и *q*".

2. Потім необхідно ввести число *x,* яке має задовольняти умову *0 < < x < q* для генерування публічного ключа *y*, натиснути кнопку "Расчет *y*".

3. Потім увести вихідний текст і натиснути кнопку "Получение хеш-кода".

4. Далі слід: увести ціле просте число *K*; натиснути кнопку "Ввод *K* "; натиснути кнопку "Расчет *r*"; натиснути кнопку "Расчет *s*"; натиснути кнопку "Сгенерировать подпись".

5. Натиснути кнопку "Конкатенация", за допомогою якої формують повідомлення.

6. Із натисканням "Проверить хеш-сумму" відбувається перевірка геш-образів вихідного та переданого текстів.

7. Далі обчислюють значення *v* і перевіряють автентичність повідомлення: для цього натискають кнопку "Верификация" (рис. 4.6).

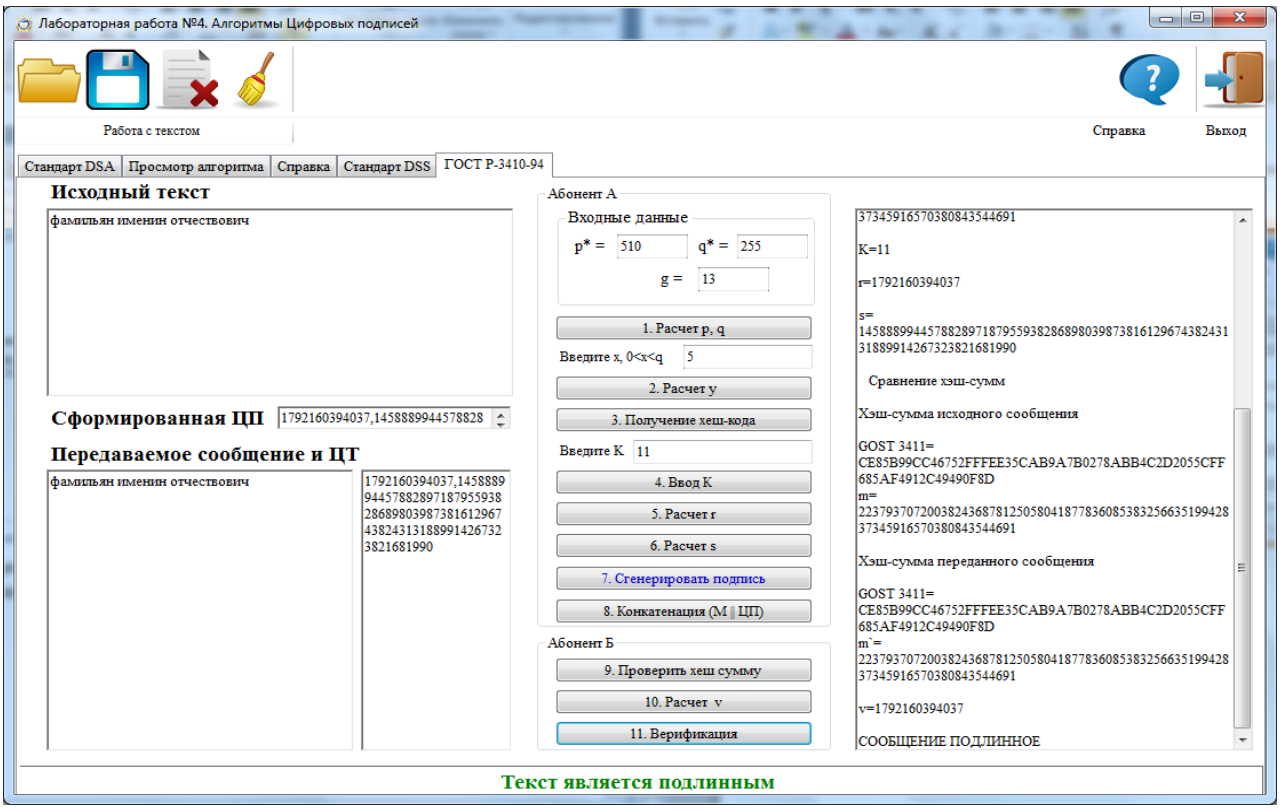


Рис. 4.6. **Формування цифрового підпису за стандартом ГОСТ Р 34.10-94**

4.5. Завдання до лабораторної роботи

Згенеруйте цифровий підпис на основі алгоритму Ель-Гамаля для відкритого тексту (як текст використовуйте ПІБ студента) (табл. 4.1).

Таблиця 4.1

Варіанти індивідуального завдання

Варіанти	Порядок простого числа *q*, бітів	Порядок простого числа *p*, бітів	Алгоритми гешування
1	2	3	4
1	896	768	Adler32
2	864	832	CRC32
3	832	896	MD2
4	800	960	MD5
5	768	1 024	SHA

Закінчення табл. 4.1

1	2	3	4
6	736	1 088	SHA256
7	704	1 152	SHA384
8	672	1 216	SHA512
9	640	1 280	Adler32
10	608	1 344	CRC32
11	576	1 408	MD2
12	544	1 472	MD5
13	512	1 536	SHA
14	480	1 600	SHA256
15	448	1 664	SHA384
16	416	1 728	SHA512
17	384	1 792	Adler32
18	352	1 856	CRC32
19	320	1 920	MD2
20	288	1 984	MD5

Підготуйте звіт установленого зразка, у висновках сформулюйте результати теоретичних досліджень і практичних умінь використання процедур формування цифрових підписів (DSS, DSA).

4.6. Контрольні запитання

1. У чому полягає сутність процедури автентифікації?
2. Які можливі загрози виникають у системі цифрового підпису. Як забезпечено захист від цих загроз?
3. Які вимоги висувають до цифрового підпису?
4. У чому відмінності підходу, використаного у DSS для створення цифрових підписів, від застосування таких алгоритмів як RSA?
5. У чому полягає сутність алгоритму цифрового підпису Ель-Гамаля?

Лабораторна робота 5
Стеганографічні методи захисту інформації

5.1. Мета

Метою цієї лабораторної роботи є закріплення теоретичного матеріалу, ознайомлення студентів з основними методами стеганографічного захисту інформації та набуття практичних навичок у використанні відповідних програмних засобів.

5.2. Рекомендації до підготовки до виконання

Для підготовки до лабораторної роботи необхідно вивчити основні поняття та визначення з питань стеганографії; ознайомитися з описом лабораторної роботи та програмним забезпеченням; підготувати відповіді на контрольні запитання.

5.3. Загальні теоретичні положення

Стеганографія – це метод організації зв'язку, який, власне, приховує сам факт передавання інформації. На відміну від криптографії, коли зловмисник точно може визначити, чи є передане повідомлення зашифрованим текстом, методи стеганографії дозволяють убудовувати секретні повідомлення у звичайні послання так, щоб неможливо було запідозрити наявність такого таємного послання.

Слово "стеганографія" у перекладі із грецької буквально означає "тайнопис" (*steganos* – секрет, таємниця; *graphy* – запис). До неї належить величезна кількість секретних засобів зв'язку: невидимі чорнила, мікрофотознімки, умовне розташування знаків, таємні канали та засоби зв'язку на "змінних" частотах тощо.

Стеганографія посідає власну нішу в інформаційній безпеці: вона не замінює, а доповнює криптографію. Приховування повідомлення методами стеганографії значно зменшує ймовірність виявлення самого факту передавання повідомлення. А якщо це повідомлення до того ж зашифроване, то воно має ще один, додатковий рівень захисту.

Сьогодні, у зв'язку з бурхливим розвитком обчислювальної техніки та нових каналів передавання інформації, з'явилися нові стеганографічні методи, в основі яких лежать особливості подавання інформації в комп'ютерних файлах, обчислювальних мережах тощо. Це дає можливість говорити про становлення нового напряму – комп'ютерної стеганографії.

Терміни та визначення

Незважаючи на те що стеганографія як спосіб приховування секретних даних відома протягом тисячоліть, комп'ютерна стеганографія – це молодий напрям, який активно розвивають.

Як і будь-який новий напрям, комп'ютерна стеганографія, незважаючи на велику кількість відкритих публікацій та щорічні конференції, довгий час не мала єдиної термінології.

Донедавна для опису моделі стеганографічної системи використовували запропоновану 1983 року Сіммонсом так звана "проблема ув'язнених".

Пізніше на конференції Information Hiding: First Information Workshop 1996 року було запропоновано використовувати єдину термінологію й обговорені основні терміни.

Стеганографічна система, або *стегосистема*, – це сукупність засобів і методів, які використовують для формування прихованого каналу передавання інформації.

Під час побудови стегосистеми потрібно враховувати такі положення:

- супротивник має повне уявлення про стеганографічну систему та деталі її реалізації. Єдиною інформацією, яка залишається невідомою потенційному супротивнику, є ключ, за допомогою якого тільки його власник може визначити факт наявності та зміст прихованого повідомлення;
- якщо супротивник яким-небудь чином довідається про факт наявності прихованого повідомлення, це не має дозволити йому знайти подібні повідомлення в інших даних, доки ключ зберігають у таємниці;
- потенційного супротивника має бути позбавлено яких-небудь технічних та інших переваг у розпізнаванні або розкритті змісту таємних повідомлень.

Узагальнену модель стегосистеми наведено на рис. 5.1.

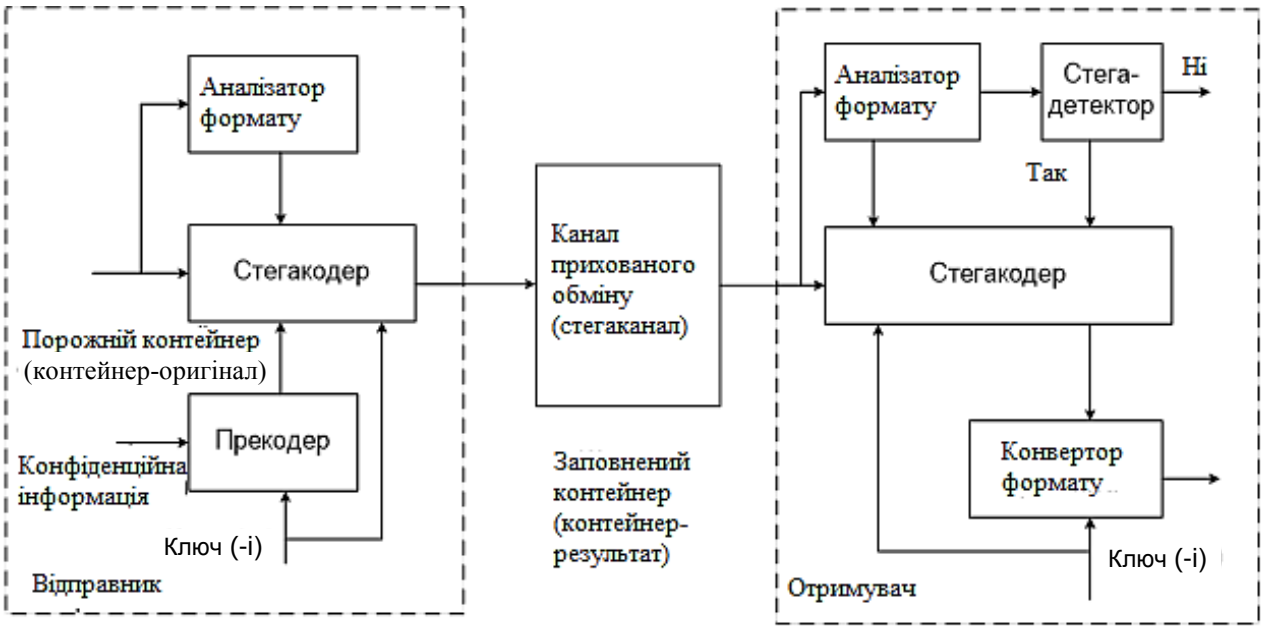


Рис. 5.1. **Узагальнена модель стегосистеми**

Такими даними може бути будь-яка інформація: текст, повідомлення, зображення тощо.

У загальному ж випадку доцільно використовувати слово "повідомлення", тому що повідомленням може бути як текст або зображення, так і, наприклад, аудіодані. Далі для позначення інформації, що приховують, будуть використовувати саме термін "повідомлення".

Контейнер – це будь-яка інформація, призначена для приховування таємних повідомлень.

Порожній контейнер – це контейнер без убудованого повідомлення; *заповнений контейнер* або *стегоконтейнер* містить убудовану інформацію.

Убудоване (*приховане) повідомлення* – повідомлення, убудоване в контейнер.

Стеганографічний канал або *стегоканал* – канал передавання стего.

Стегоключ або просто *ключ* – це секретний ключ, необхідний для приховування інформації. Залежно від кількості рівнів захисту (наприклад, убудовування зашифрованого повідомлення) у стегосистемі, може бути один або декілька стегоключів.

Аналогічно до криптографії, за типом стегоключа стегосистеми можна розподілити на два типи:

- із секретним ключем;
- із відкритим ключем.

У стегосистемі із секретним ключем використовують один ключ, який має бути визначено або до початку обміну секретними повідомленнями, або переданий захищеним каналом.

У стегосистемі з відкритим ключем для вбудовування та вилучення повідомлення використовують різні ключі, які відрізняються таким чином, що за допомогою обчислень неможливо отримати один ключ із іншого. Тому один ключ (відкритий) можна передавати вільно незахищеним каналом зв'язку. Ця схема добре працює в разі взаємної недовіри сторін.

Вимоги до стегосистеми. Будь-яка стегосистема має відповідати таким вимогам:

- властивості контейнера може бути модифіковано так, щоб зміну неможливо було виявити візуальним контролем. Ця вимога визначає якість убудовування повідомлення: воно жодним чином не має привернути увагу зловмисника;
- стегоповідомлення має бути стійким до перетворень, зокрема й зловмисним. У процесі передавання зображення (звук або інший контейнер) може зазнавати різних трансформацій: зменшуватися або збільшуватися, перетворюватися на інший формат тощо. Крім того, воно може стискатися, зокрема й із використанням алгоритмів стиску зі втратою даних;
- для збереження цілісності вбудованого повідомлення необхідно використовувати завадостійке кодування;
- для підвищення надійності повідомлення, що вбудовують, мають дублювати.

Застосування стеганографії

Сьогодні можна виділити три тісно пов'язані напрями застосування стеганографії, що мають одне коріння: приховування даних (повідомлень), цифрові водяні знаки та заголовки (рис. 5.2).

Рис. 5.2. **Застосування цифрової стеганографії**

Приховування даних, часто значного обсягу, висуває серйозні вимоги до контейнера: його розмір має перевищувати розмір даних, що приховують, у кілька разів.

Цифрові водяні знаки використовують для захисту авторських або майнових прав на цифрові зображення, фотографії або інші оцифровані твори мистецтва. Основні вимоги, які ставлять до таких убудованих даних, є надійність і стійкість до перетворень.

Цифрові водяні знаки мають невеликий обсяг, але з урахуванням зазначених вимог для їхнього вбудовування використовують більш складні методи, ніж для вбудовування повідомлень або заголовків.

Заголовки використовують переважно для маркування зображень у великих електронних сховищах (бібліотеках) цифрових зображень, аудіо- та відеофайлів.

У цьому разі стеганографічні методи використовують не тільки для впровадження ідентифікаційного заголовку, але й інших індивідуальних ознак файлу.

Заголовки, що впроваджують, мають невеликий розмір, а вимоги до них мінімальні: вони не мають уносити значних змін, проте бути стійкими до основних геометричних перетворень.

Обмеження

Кожне із зазначених застосувань потребує певного співвідношення між стійкістю вбудованого повідомлення до зовнішніх впливів (зокрема й стегоаналізу) і розміром повідомлення, що вбудовують.

Для більшості сучасних методів має місце така залежність надійності системи від обсягу вбудованих даних (рис. 5.3).

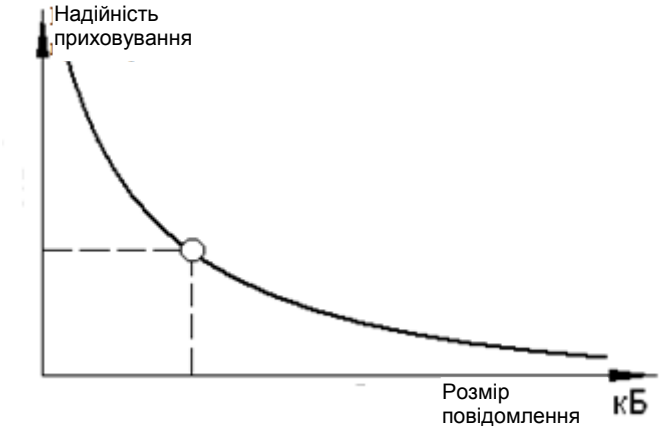


Рис. 5.3. **Графік залежності надійності системи від обсягу даних, що вбудовують**

Ця залежність показує, що зі збільшенням обсягу даних, що вбудовують, знижується надійність системи (за незмінності розміру контейнера). Таким чином, використаний у стегосистемі контейнер накладає обмеження на розмір убудованих даних.

Контейнери

Істотний вплив на надійність стегосистеми та можливість виявлення факту передавання прихованого повідомлення має вибір контейнера. *Наприклад*, досвідчене око цензора з художньою освітою легко виявить зміну гами кольорів у репродукціях "Мадонни" Рафаеля або "Чорного квадрата" Малевича із вбудованим повідомленням.

За довжиною контейнери можна розподілити на два типи: безперервні (потокові) й обмеженої (фіксованої) довжини. Особливістю *потокового контейнера* є те, що неможливо визначити його початок або кінець. Більш того, немає можливості довідатися заздалегідь, якими будуть наступні шумові біти. Це призводить до необхідності включати біти, що приховують повідомлення, у потік у реальному масштабі часу. Самі ж приховувальні біти вибирають за допомогою спеціального генератора, який задає відстань між послідовними бітами в потоці.

У безперервному потоці даних найбільші труднощі для отримувача – визначити, коли починається приховане повідомлення. За наявності в потоковому контейнері сигналів синхронізації або меж пакета, повідомлення починається відразу після одного з них. Своєю чергою, для відправника можливі проблеми, якщо він не впевнений у тому, що потік контейнера буде досить довгим для розміщення цілого таємного повідомлення.

За використання *контейнерів фіксованої довжини* відправник заздалегідь знає розмір файлу та може вибрати приховувальні біти в потрібній псевдовипадковій послідовності. З іншого боку, контейнери фіксованої довжини, як це вже зазначалося, можуть умістити обмежене повідомлення, яке іноді може не помістится у файл-контейнер.

Інший недолік полягає в тому, що відстані між приховувальними бітами рівномірно розподілено між найбільш короткою і найбільш довгою заданими відстанями, тоді як дійсний випадковий шум буде мати експонентний розподіл довжин інтервалу. Звичайно, можна згенерувати псевдовипадкові числа з експонентним розподілом, але цей шлях є занадто складним. Однак на практиці найчастіше використовують саме контейнери фіксованої довжини як найпоширеніші та доступні.

Можливі такі варіанти контейнерів:

• контейнер генерує сама стегосистема. Прикладом є програма Mandelsteg, де як контейнер для вбудовування повідомлення генерують фрактал Мандельброта. Такий підхід можна назвати *конструкційною стеганографією*;

• контейнер вибирають з певної множини контейнерів. У цьому разі генерують велику кількість альтернативних контейнерів, щоб потім вибрати найбільш придатний для приховування повідомлення. Такий підхід можна назвати *вибірковою стеганографією*. У цьому разі під час вибору оптимального із множини згенерованих контейнерів найважливішою вимогою є його природність. Єдиною проблемою залишається те, що навіть оптимально організований контейнер дозволяє приховати незначну кількість даних за значного розміру самого контейнера;

• контейнер надходить іззовні. У цьому разі відсутня можливість його вибору, отже, він не завжди може ідеально підходити до повідомлення, що вбудовують. Називають це *безальтернативною стеганографією*.

Методи приховування інформації

Найпоширенішим, але найменш стійким є метод заміни найменш значущих бітів або LSB-метод. Він полягає у використанні похибок дискретизації, які завжди наявні в оцифрованих зображеннях або аудіо- і відеофайлах. Ці похибки дорівнюють найменшому значущому розряду числа, яке визначає глибину кольору елемента зображення (пікселя). Тому модифікація молодших бітів у здебільшого не викликає значної трансформації зображення та її не виявляють візуально. Докладніше цей метод описано у статті В. М. Кустова й О. О. Федчука "Методи вбудовування прихованих повідомлень".

Іншим популярним методом убудовування повідомлень є використання особливостей форматів даних, які використовують стиск із втратою даних (наприклад, *JPEG*). Цей метод (на відміну від *LSB*) стійкіший до геометричних перетворень і виявлення каналу передавання, тому що є можливість у широкому діапазоні варіювати якість стиснутого зображення, що унеможливлює визначення походження викривлення.

Для вбудовування цифрових водяних знаків використовують більш складні методи.

Метод заміни найменш значущого біта

Метод заміни найменшого значущого біта (НЗБ, LSB – Least Significant Bit) найпоширеніший серед методів заміни у просторовій множині. Молодший значущий біт зображення містить найменше інформації. Відомо, що людина здебільшого не здатна помітити змін у цьому біті. Фактично НЗБ – це шум, тому його можна використовувати для вбудовування інформації заміною найменш значущих бітів пікселей зображення бітами секретного повідомлення. Водночас для зображення у градаціях сірого (кожний піксель зображення кодують одним байтом) обсяг убудованих даних може становити 1/8 від загального обсягу контейнера.

Наприклад, у зображення розміром 512 × 512 можна вмонтувати ~32 кБ інформації. Якщо ж модифікувати два молодші біти (що також практично непомітно), то цю пропускну спроможність можна збільшити ще вдвічі.

Популярність цього методу обумовлено його простотою та тим, що він дозволяє приховувати у відносно невеликих файлах досить великі обсяги інформації (пропускна спроможність створюваного прихованого каналу зв'язку становить від 12,5 до 30 %). Метод найчастіше працює з растровими зображеннями у форматі без компресії (наприклад, GIF і BMP).

Метод НЗБ має низьку стеганографічну стійкість до атак пасивного й активного порушників. Основний його недолік – висока чутливість до найменших перетворень контейнера. Для зменшення цієї чутливості часто додатково застосовують завадостійке кодування.

Перед імпортом зображення-контейнера в документ MathCAD, його необхідно підготувати у відповідному редакторі та записати у вигляді файлу в поточний (для документа MathCAD) каталог. Слід зазначити: для уникнення можливих проблем із підтриманням кирилиці бажано, щоб адреса розміщення файлу на диску, як, власне, й ім'я файлу, складалися з латинських символів. MathCAD підтримує формати BMP, JPEG, GIF, PCX і TGA. Як було зазначено, формати BMP і GIF дозволяють зберігати зображення практично без утрати їхньої якості, тому більш придатні як носії інформації.

Крок 1. Розгляньте структуру BMP-файлу: він містить точкове (растрове) зображення та складається із трьох основних розділів: заголовка файлу, заголовка растру та растрових даних.

Заголовок файлу містить інформацію про файл (його тип, розмір тощо).

У *заголовок растру* винесено інформацію про ширину та висоту зображення, кількість бітів на піксель, розмір растру, глибину кольору, коефіцієнт компресії тощо.

Насамперед нас цікавлять растрові дані – інформація про колір кожного пікселя зображення. Колір пікселя визначено об'єднанням трьох основних колірних складових частин: червоного, зеленого та синього кольорів (скорочено, RGB). Кожному з них відповідає своє значення інтенсивності, яке може змінюватися від 0 до 255. Отже, за кожний із каналів відповідає 8 бітів (1 байт), а глибина кольору зображення загалом – 24 біти (3 байти).

Імпорт графічного файлу виконується операцією Picture з позиції Insert головного меню програми. У модулі, що з'явився, необхідно заповнити шаблон даних у лівому нижньому куті: у подвійних лапках треба ввести ім'я файлу (або ж, за потреби, повний шлях його розміщення на диску) і натиснути клавішу <Enter>.

Формати BMP і GIF використовують алгоритми компресії, але ці алгоритми найпростіші, що дозволяє зберігати зображення практично без утрати його якості. Для цього застосовують функцію READRGB ("ім'я файлу"), що повертає масив із трьох підмасивів, які, своєю чергою, дають інформацію про розкладання кольорового зображення на компоненти *R, G* і *B*:

C := READRGB("C.bmp").

Водночас три компоненти кольорів розміщаються один за одним у загальному масиві *C*.

Для виділення складових частин кольору можна використовувати вбудовані функції виділення відповідних компонентів, кожна з яких повертає масив, що відповідає визначеному компоненту кольору графічного файлу:

R := READ_RED("C.bmp"); G := READ_GREEN("C.bmp");
B := READ_BLUE("C bmp").

Крок 2. Текст повідомлення збережіть у файлі M.txt каталогу, що перебуває у тій самій папці, що й документ MathCAD, у такому форматі:

тип файлу – звичайний текст (*.txt);

кодування – кирилиця (Windows).

Крок 3. Є можливість приховування файлів будь-якого формату. Єдина умова – вибір файлу-контейнера належного обсягу.

Імпорт текстового повідомлення можна виконати за допомогою функції READBIN ("ім'я_файлу", "тип_формату_даних"). У цьому разі дані подано як 8-бітне беззнакове ціле число (байтів):

M := READBIN("M.txt", "byte").

Результат обчислення цього виразу – це матриця-стовпець (вектор), кожний елемент якої відповідає розширеному ASCII-коду відповідного символу (букви) імпортованого повідомлення. У десятковому вигляді коди символів можуть набувати значень від 0 до 255; у двійковому вигляді для цього досить використовувати 8 бітів на один символ, так зване однобайтове кодування, на що вказує параметр byte як аргумент функції READBIN.

Необхідно зазначити, що за замовчуванням нижня межа індексації масивів дорівнює 0. У прикладах індексація починається з 1. Це, зокрема, можна встановити за допомогою оператора *ORIGIN :=1* на початку документа або ввести 1 у поле Array Origin на вкладці Built-in Variables діалогового вікна Worksheet Options, що викликають із меню Tools системи MathCAD.

Перевірку імпортування файлу повідомлення (коли як повідомлення використовують звичайний текст) можна виконати за допомогою виклику функції *vec2str*(*M*). Ця функція повертає рядок символів, які відповідають вектору MASCII-кодів.

Крок 4. Для того щоб під час розпакування контейнера з визначеної множини символів можна було чітко знайти початок і кінець прихованого повідомлення, доцільно ввести відповідні секретні мітки, які обмежували б його.

Мітки мають складатися з достатньої кількості символів, щоб не вважати за них випадкові символи. Крім того, для зменшення ймовірності виявлення міток під час здійснення стеганоаналізу бажано, щоб коди цих

символів було достатньо рознесено на ASCII-осі (наприклад, використовувати поряд із латинськими символами символи кирилиці та службові символи – так звану транслітерацію; використання псевдовипадкових послідовностей кодів символів і т. ін.). Нехай мітки мають такий вигляд:

$$\mu_S = \text{"n0ч@m0k"} \text{ і } \mu_E = \text{"KIHeu,6"}.$$

Обмежувальні мітки додають у текст закодованого повідомлення, для чого використовують функцію *stack*(*A, B, ...*), що дозволяє поєднувати записані через кому масиви. Об'єднання відбувається шляхом "насадження" матриці *A* на матрицю *B*; визначену в такий спосіб матрицю – на наступну матрицю (якщо така є) і т. д.

Зрозуміло, що початкові матриці повинні мати однакову кількість стовпців, тому необхідно перетворити мітки з рядків на вектори *ASCII*-кодів. Отже,

$$sMe :stack\ (str2vec(\mu_S),\ M\_cod,\ str2vec(\mu_E)).$$

Загальна кількість символів у прихованому повідомленні:

$$rows(sMe) = 5\ 404.$$

Кількість НЗБ контейнера, для цього необхідно (8 бітів/символ):

$$8 \times rows(sMe) = 43\ 232\ (біти).$$

Загальна кількість НЗБ контейнера:

$$rows(C)\ cols(C) = 3 \cdot 128 \cdot 128 = 49\ 152 > 43\ 232\ (біти).$$

Таким чином, файл зображення має достатній обсяг для того, щоб приховати повідомлення.

Крок 5. Для подальших обчислень буде потрібно перетворення десяткового числа (яким за замовчуванням кодують кожний символ) на формат двійкового. Також знадобиться і зворотне перетворення.

Зворотний процес, тобто отримання повідомлення із зображення, здійснюють зворотним шляхом, а саме: визначенням молодших бітів зображення та перетворення їх на повідомлення.

Цифрові водяні знаки

У сучасних системах формування цифрових водяних знаків використовують принцип убудовування мітки, яка є вузькосмуговим сигналом у широкому діапазоні частот маркованого зображення. Зазначений метод реалізують за допомогою двох різних алгоритмів та їхніх можливих модифікацій. У першому випадку інформацію приховують шляхом фазової модуляції інформаційного сигналу (носієвої) із псевдовипадковою послідовністю чисел. У другому – наявний діапазон частот розподіляють на кілька каналів і передавання здійснюють між цими каналами. Щодо вихідного зображення, мітка є деяким додатковим шумом, але тому, що шум у сигналі наявний завжди, його незначне зростання за рахунок упровадження мітки не дає помітних для ока модифікацій. Крім того, мітка розсіюється по всьому вихідному зображенню, у результаті чого стає більш стійкою щодо вилучення.

Сьогодні комп'ютерну стеганографію продовжують розвивати: формують теоретичну базу, розробляють нові, більш стійкі методи вбудовування повідомлень. Серед основних причин сплеску інтересу до стеганографії можна виділити взяті в ряді країн обмеження на використання сильної криптографії, а також проблему захисту авторських прав на художні твори в цифрових глобальних мережах.

Історичні замітки

Історія стеганографії – це історія розвитку людства.

Місцем народження стеганографії багато фахівців називають Єгипет, хоча першими стеганографічними повідомленнями можна назвати й наскельні малюнки прадавніх людей.

Перше згадування про стеганографічні методи в літературі приписують Геродоту. Історик описав випадок передавання повідомлення Демартом, який зіскрібав віск із дощечок, писав листи прямо на дереві, а потім заново покривав дощечки воском.

Інший епізод, який зараховують до тих самих часів, – передавання послання з використанням голови раба. Для цього голову раба голили, наносили на шкіру татуювання і, коли волосся відростало, відправляли з посланням.

У Китаї листи писали на смужках шовку. Для приховування повідомлень смужки з текстом звивали в кульки, покривалися воском і потім посланці ковтали їх.

Темне Середньовіччя породило не тільки інквізицію: посилення стеження призвело до розвитку як криптографії, так і стеганографії. Саме в Середні віки вперше було застосовано спільне використання шифрів і стеганографічних методів.

У XV ст. чернець Трітеміус (1462 – 1516 рр.), який займався криптографією та стеганографією, описав багато різних методів прихованого передавання повідомлень. 1499 року, ці записи об'єднали у книгу Steganographia, яку сьогодні, знаючи латину, можна прочитати у мережі "Інтернет".

XVII – XVIII ст. відомі як ера "чорних кабінетів" – спеціальних державних органів із перехоплення, перлюстрації та дешифрування листування. До штату "чорних кабінетів", крім криптографів і дешифрувальників, входили й інші фахівці, зокрема хіміки. Наявність фахівців-хіміків було необхідне через активне використання так званого невидимого чорнила. Прикладом може слугувати цікавий історичний епізод: повсталими дворянами в Бордо було заарештовано францисканського ченця Берто, який був агентом кардинала Мазаріні. Повстанці дозволили Берто написати лист знайомому священикові в місто Блей. Однак наприкінці цього листа релігійного змісту чернець зробив приписку, на яку ніхто не звернув увагу: "Посилаю Вам очну мазь; натріть нею очі, й Ви будете краще бачити". Так він зумів переслати не тільки приховане повідомлення, але й указав спосіб його виявлення. У результаті чернця Берто було врятовано.

Стеганографічні методи активно використовували й у роки громадської війни між жителями Півдня та Півночі Америки. Так, 1779 року два агенти жителів Півночі Семюель Вудхолл і Роберт Таунсенд передавали інформацію Джорджу Вашингтону, використовуючи спеціальне чорнило.

Різне симпатичне чорнило використовували й російські революціонери на початку XX ст., що знайшло відображення в радянській літературі: Куканов у повісті "У истоков грядущего" описує застосування молока як чорнила для написання таємних повідомлень. Утім, царська охоронка теж знала цей метод (в архіві зберігають документ, у якому описано спосіб використання симпатичного чорнила та наведено текст перехопленого таємного повідомлення революціонерів).

Особливе місце в історії стеганографії посідають фотографічні мікрокрапки. Так, ті самі мікрокрапки, які зводили з розуму спецслужби США під час Другої світової війни. Однак мікрокрапки виникли набагато раніше,

відразу ж після винаходу Дагером фотографічного процесу, і вперше у військовій справі їх було використано за часів франко-пруської війни (1870 року).

Робота зі SteganosSecuritySuite 15

Усупереч крилатому твердженням про те, що комп'ютер ніколи не помиляється, цілком довіряти йому свої секрети не варто. Інформація на жорсткому диску вашого домашнього або робочого комп'ютера – той самий щоденник, у якому ви в подробицях викладаєте про все, що відбувалося з вами за день. Діставши доступ до цього комп'ютера, за вашими "слідами" роботи на ньому, недоброзичливець може витягти багато корисної інформації: від листів у вашій поштовій скриньці до даних про кредитну картку. Щоб цього не сталося, потрібно дотримуватися кількох простих правил:

- нікого не посвячувати в те, які паролі ви використовуєте;
- не тримати паролі на папері поруч зі своїм робочим місцем;
- використовувати антивірусне програмне забезпечення;
- і, нарешті, мати під рукою набір інструментів для маскування важливих даних і знищення будь-яких слідів роботи за комп'ютером.

Для останнього правила цілком підійде пакет утиліт Steganos Security Suite 15, який містить дев'ять інструментів "на всі випадки життя" (рис. 5.4).

Рис. 5.4. **Steganos Privacy Suite 15**

Сейф. Документи особливої важливості, як відомо, зберігають у сейфі. Для електронних документів розробники Steganos Privacy Suite 2015, за аналогією, створили віртуальний сейф. Принцип його дії полягає в такому: на одному із жорстких дисків виділяють область, яку використовують для створення віртуального диска. Кількість віртуальних дисків може бути довільною. Після виділення місця в системі з'являється додатковий диск, ніби ви підключили модуль пам'яті USB або додатковий вінчестер (рис. 5.5).

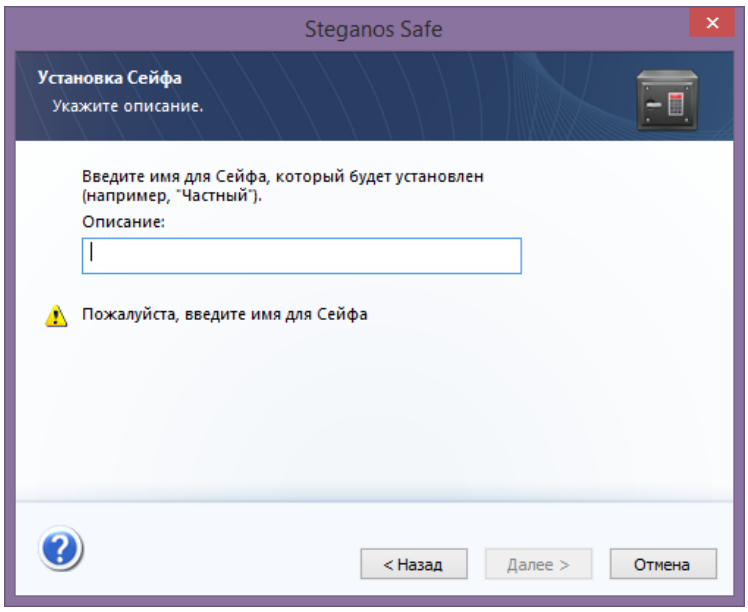


Рис. 5.5. **Уведення імені сейфа**

Кожен віртуальний диск може зберігати до 256 ГБ даних за використання файлової системи NTFS і до 4 ГБ – на файловій системі FAT32. Такий віртуальний диск призначено для того, щоб зберігати на ньому найважливіші документи. Отже, можна не турбуватися про те, що хтось, окрім вас, прочитає цю інформацію. По-перше, усі дані, що записують на диск, шифрують "на льоту" з використанням стійкого алгоритму шифрування, а по-друге, після того, як роботу з диском буде завершено, його можна (і навіть потрібно) приховати, завершивши попередньо роботу з усіма документами на віртуальному пристрої.

В останній версії програми подано цікаву можливість використання як пароль послідовності зображень (рис. 5.6). Програма пропонує 36 зображень (це можуть бути фотографії або символи), із яких потрібно вибрати три. Потрібно запам'ятати не тільки те, які картинки вибрали, але й їхню послідовність.

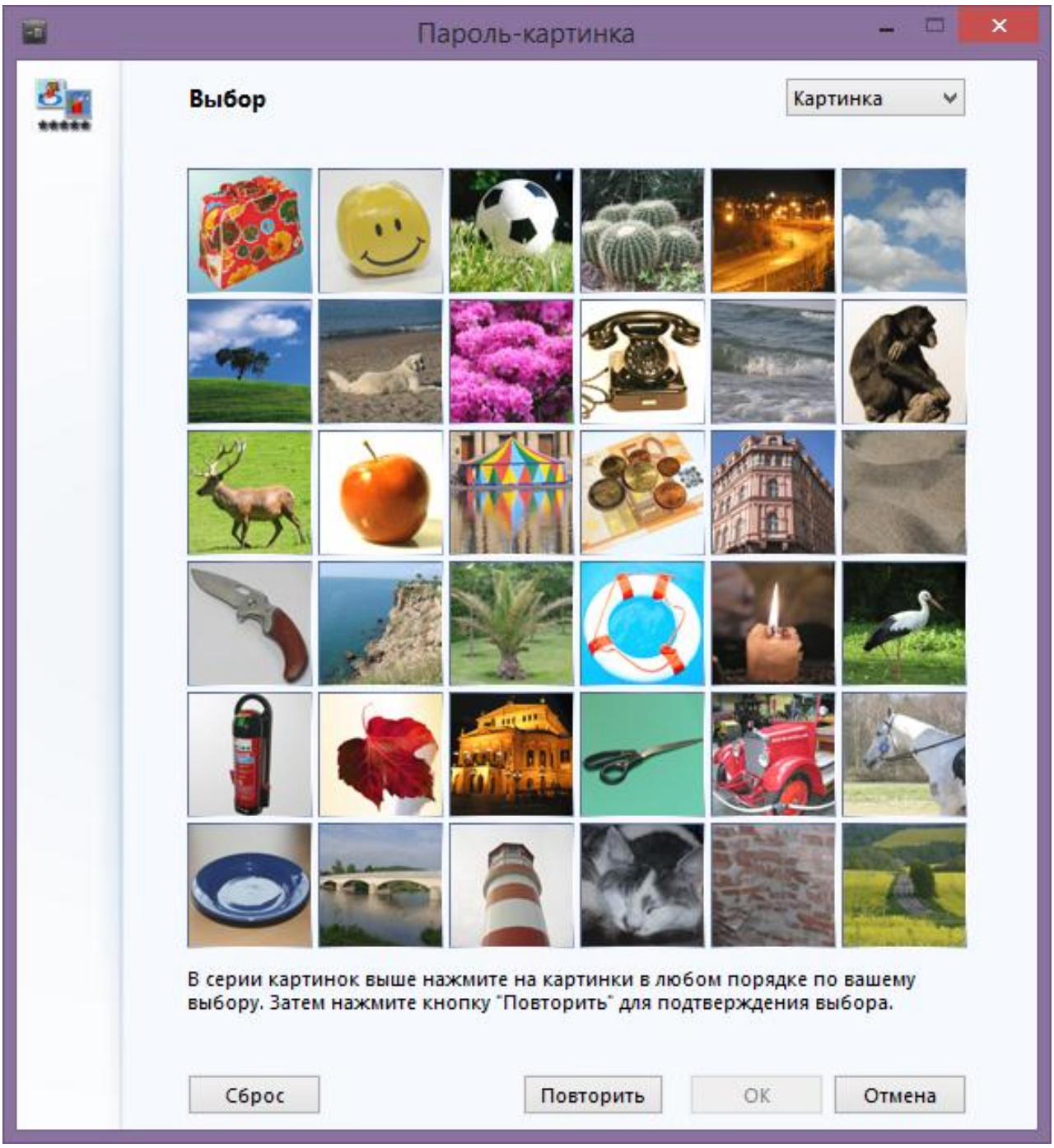


Рис. 5.6. **Пароль-картинка**

Як пароль можна також використовувати iPod. У цьому разі не потрібно щось запам'ятовувати, і найголовніше, не втрачають плеєр, який слугує ключем до відкриття віртуального сейфа.

Для швидкого відкриття створеного сейфа можна використовувати певну комбінацію клавіш (рис. 5.7). Правда, можливості самостійного призначення такої комбінації немає, можна лише вибрати найзручніший для вас із запропонованого програмою списку.

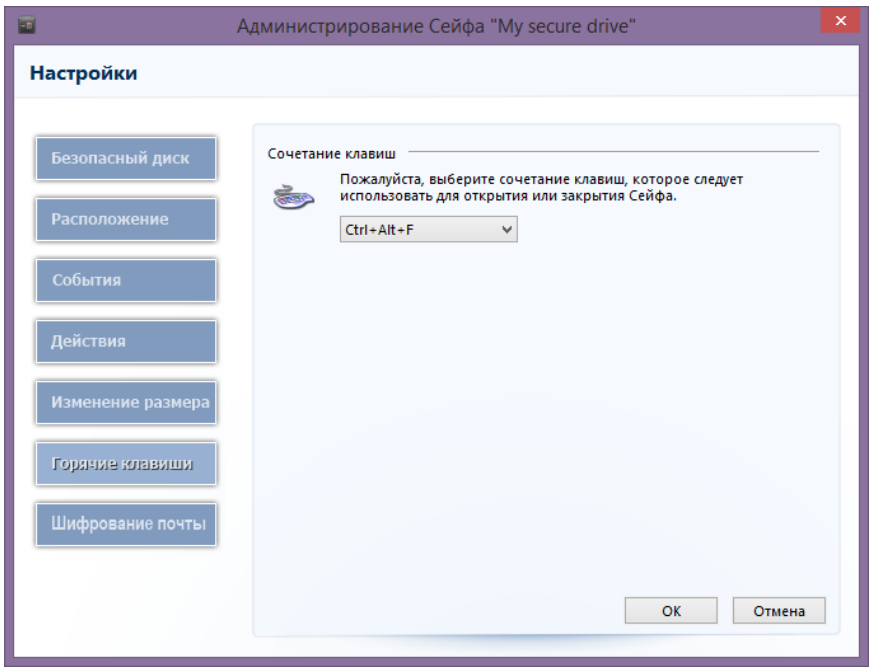


Рис. 5.7. **Налаштування сейфа**

Портативний сейф. Утиліта Steganos Privacy Suite *15* забезпечує збереження даних зі зберіганням інформації на портативних цифрових носіях (рис. 5.8).

Наприклад, її можна використовувати для USB-модулів пам'яті або для запису CD/DVD-дисків, DVD-DL, Blu-Ray або Blu-Ray-DL. Причому для читання на комп'ютері даних із носіїв, захищених Portable Safe, не потрібно, щоб на комп'ютері було встановлено цю програму, досить підключити носій до ПК або вкласти у пристрій читання.

Рис. 5.8. **Вибір місця зберігання портативного сейфа**

Приватне вибране. У міру того як зростає кількість відвіданих сайтів в інтернеті, неодмінно збільшується список закладок. Їх видно будь-кому, хто має доступ до вашого комп'ютера. Небажано, щоб стороння особа могла побачити ці посилання. Їх можна "сховати", використовуючи цю утиліту. Така утиліта сподобається тим, хто на роботі ділить комп'ютер ще з одним або навіть кількома людьми. Вона дозволяє використовувати для зберігання посилань, крім стандартної папки Favorities, ще одну папку, яку називають Private Favorites. Дістати доступ до закладок із цієї папки можна тільки після введення правильного пароля (рис. 5.9).

Рис. 5.9. **Приватне вибране**

Менеджер паролів. Цю просту утиліту призначено для зберігання паролів для входу на різні сайти, які потребують авторизації, для збереження даних, що потрібно вводити для оплати в інтернет-магазинах, та іншої конфіденційної інформації. Для роботи із програмою достатньо запам'ятати один пароль, який будуть використовувати для доступу до неї. Навіть якщо програму весь час відкрито, пароль потрібно буде вводити знову через певні проміжки часу. Таким чином програма захищає дані від можливої крадіжки третіми особами, які можуть сісти за комп'ютер замість вас.

Щоб не плутатися в численних паролях, їх можна розподілити за категоріями, а також для кожного ввести свій опис. Якщо хочете зберегти логін і пароль для входу в захищений розділ сайту, то можна ввести його адресу в поле URL. Далі для відкриття сайту у браузері достатньо буде натиснути кнопку у вікні програми (рис. 5.10).

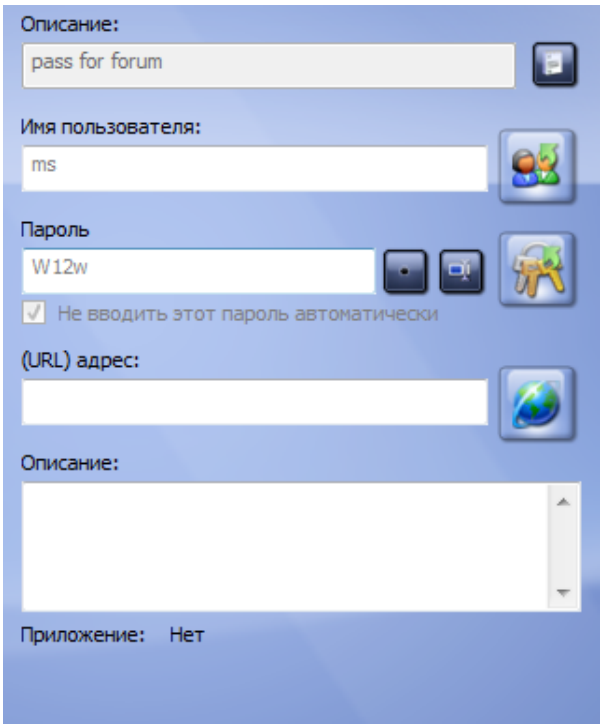


Рис. 5.10. **Менеджер паролів**

У Steganos Security Suite 15 є можливість синхронізації паролів із портативними пристроями, що працюють на базі Windows CE. Для синхронізації із КПК досить натиснути кнопку "Синхронизация" на панелі інструментів програми.

Шифрування пошти. Незважаючи на те що електронна пошта здається досить захищеним засобом зв'язку, насправді це не так. Лист цілком можуть перехопити на шляху від відправника до отримувача, особливо якщо мова йде про важливі дані. Якщо ведете листування, яке може зацікавити зловмисників, то має сенс скористатися програмою E-Mail Encryption. Вона дає можливість зашифрувати текст листа, щоб навіть у разі перехоплення його вміст залишився для недоброзичливця таємницею.

Робота із програмою може відбуватися в одному із двох режимів: можна вводити текст листа безпосередньо у вікні програми або ж

використовувати для шифрування кнопку Encrypt, яку додають на панель інструментів поштових клієнтів (наприклад, Outlook Express).

Якщо використовуєте поштову програму, яка не підтримується E-Mail Encryption, достатньо ввести текст листа у вікні програми для шифрування та зберегти текст у вигляді файлу, який потім відправити як укладення. Шифрувати можна не тільки текст, а також файли та навіть цілі папки. Щоб розшифрувати вміст повідомлення, отримувач має знати тільки пароль, наявність на його комп'ютері E-Mail Encryption не обов'язкова (рис. 5.11).

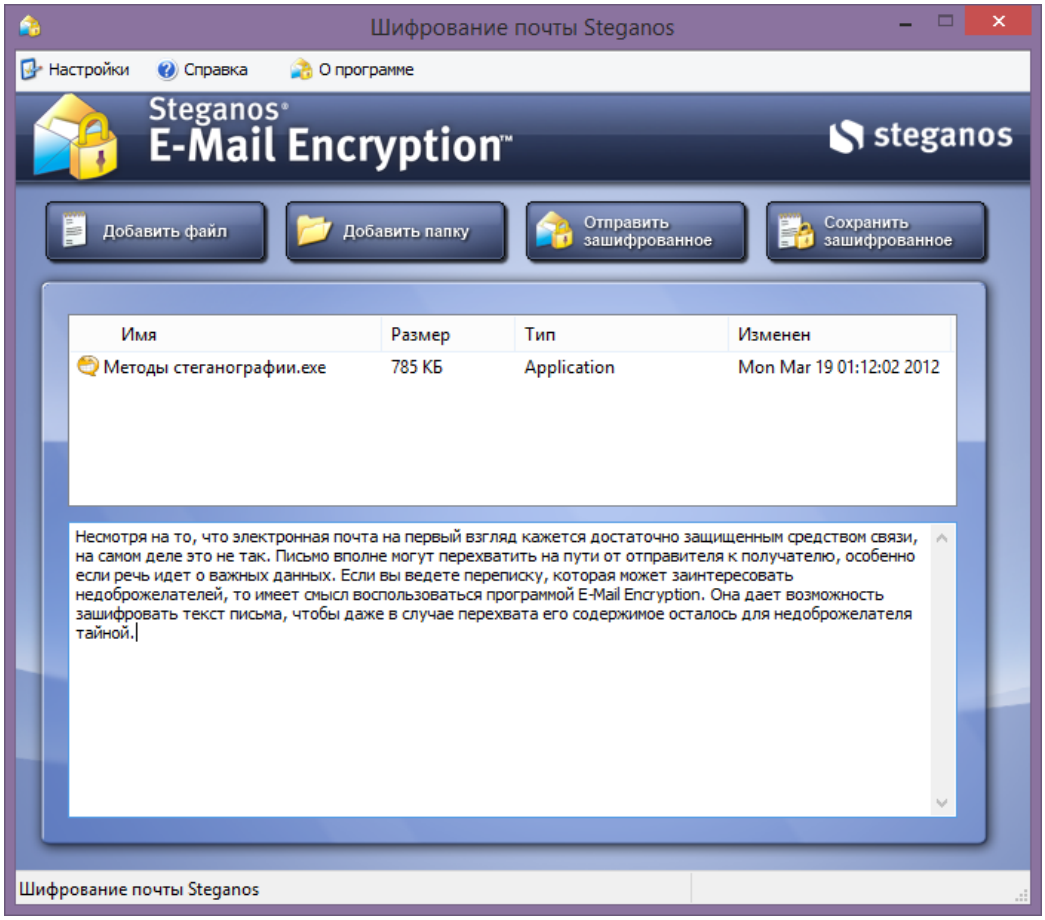


Рис. 5.11. **Шифрування пошти**

Шифрування&Приховування. Цю утиліту призначено для шифрування вмісту файлів і папок, а також для їхнього приховування. Якщо вирішите просто зашифрувати файли та папки, програма створить файл із розширенням .sef, для відкриття якого необхідно буде ввести пароль. Під час створення такого файла програма попросить указати, чи потрібно видаляти вихідні файли із жорсткого диска або ж їх можна зберегти.

Але функція приховування даних, яку також реалізовано в File Manager, є набагато цікавішою. Використовуючи її, можна не просто зашифрувати файли, але й заховати їх у якому-небудь графічному або аудіофайлі. Водночас у файловому менеджері буде відображено тільки файл-прикриття, а справжній уміст графічного файлу можна подивитися, тільки відкривши його в "Шифрування & Приховування" та ввівши пароль (рис. 5.12).

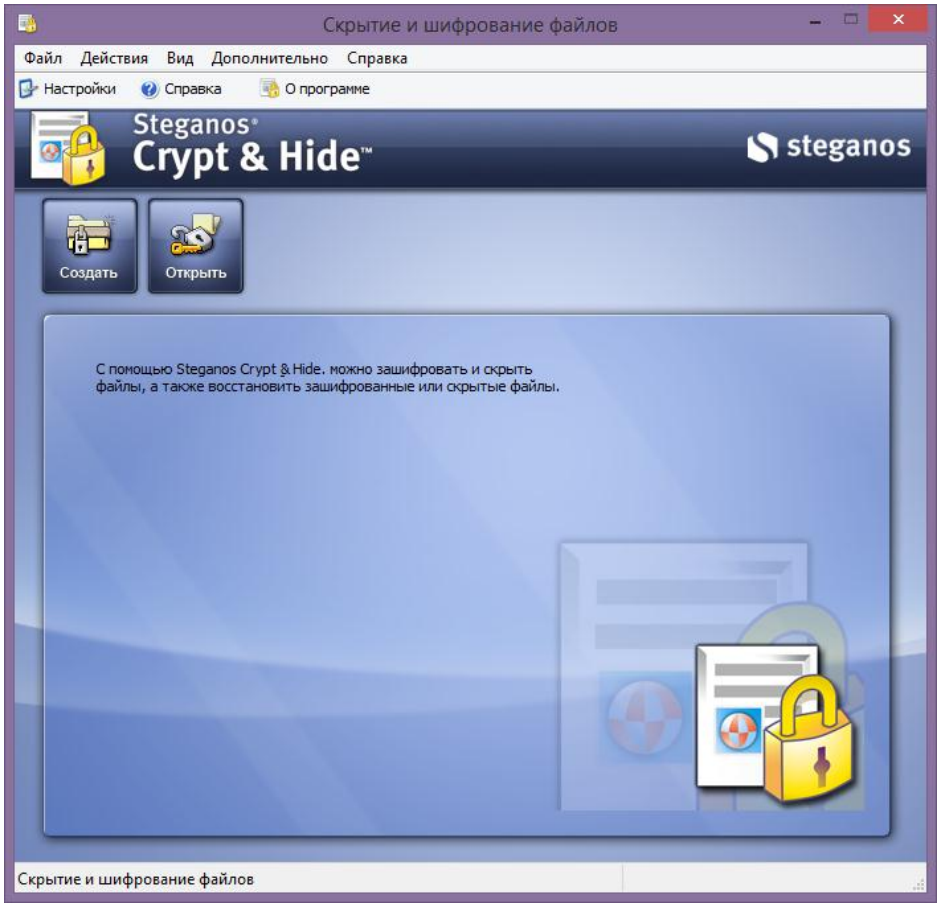


Рис. 5.12. **Приховування та шифрування файлів**

Ця функція є дуже зручною, оскільки файли з розширенням **.edf можуть привернути увагу, а нешкідливі картинки або файли MP3 – ні.

Зачищення слідів. Якщо для вас важливо, щоб людина, яка буде працювати за комп'ютером після вас, не змогла подивитися, які сайти ви відвідували, яку музику слухали та які файли відкривали, скористайтеся утилітою TraceDestructor. Ця програма дозволяє видалити тимчасові файли, список файлів, які нещодавно відкривали, список програм, які нещодавно запускали, файли cookies і History браузера Internet Explorer і багато іншого. Вона працює з дуже багатьма програмами й на першу

вимогу видаляє всі дані, які можуть вас скомпрометувати. Серед підтримуваних програм – менеджери завантаження (GetRight, Download Accelerator), інтернет-пейджери (ICQ, Trillian, Miranda), пірингові клієнти (eMule, Bittorent тощо), програвачі мультимедійних файлів (усі версії Windows Media Player аж до 11, Quicktime, WinAmp, Realplayer) і багато інших програм, зокрема навіть популярні переглядачі графічних файлів.

У цій же програмі можна відключити деякі функції Windows, які можуть бути загрозою для безпеки, наприклад відключити службу Messenger, відсилання логів про виниклі помилки в Microsoft, відсилання даних із Media Player, роботу Windows Firewall тощо (рис. 5.13).

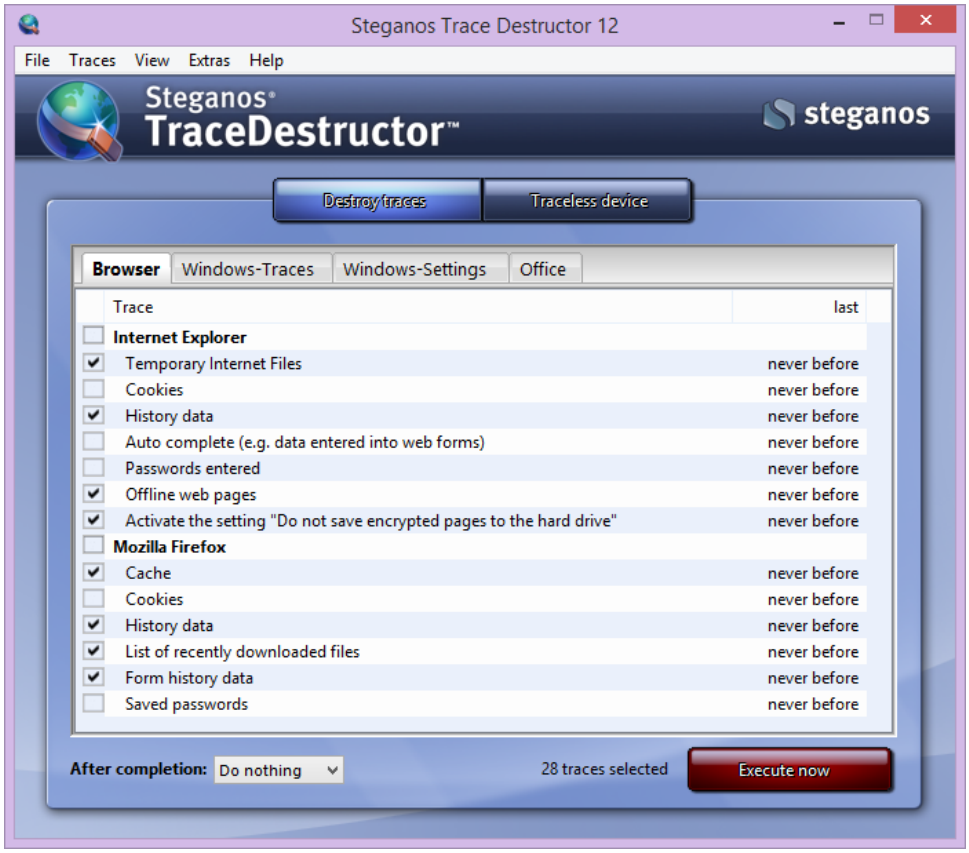


Рис. 5.13. **Зачищення слідів**

Шредер. Програму пакета Steganos Security Suite 15 призначено для повного видалення даних із жорсткого диска. Як відомо, після стандартного видалення файлів і папок із вінчестера дані на ньому все одно залишаються. Їх можна з легкістю відновити, використовуючи спеціальні програми. Якщо використовувати для видалення *Шредер*, можна бути повністю впевненим у тому, що жодна програма для відновлення не зможе

показати наявність видалених файлів та їхній уміст. Для видалення даних програма може використовувати одну із трьох технологій, кожна з яких є достатньо надійною. Наприклад, один із запропонованих методів видалення використовують у міністерстві оборони США і передбачає кількаразове перезаписування даних.

Якщо, установивши *Шредер*, пошкодували про те, що не використали програму раніше, і турбуєтеся, що видалені раніше файли можна відновити, можна використовувати інструмент для глибокого очищення жорсткого диска. Він надійно видалить усі дані, які можуть залишатися в тій області диска, яку визначено системою як вільну (рис. 5.14).

Рис. 5.14. **Шредер**

Стерти в один клік. Усі заздалегідь вибрані сліди в *Зачищенні слідів* можна видаляти за допомогою одного кліка "Стерти в один клік" (рис. 5.15).

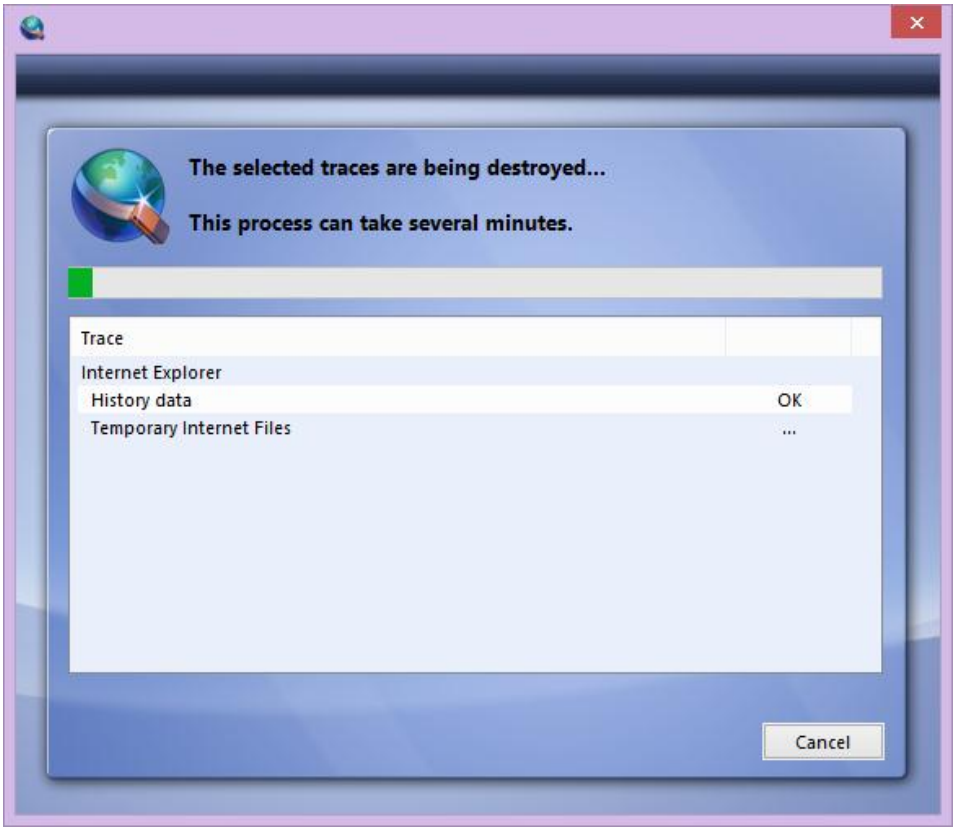


Рис. 5.15. **Стерти в один клік**

Висновок. Припускати, що коли-небудь буде винайдено універсальний спосіб боротьби з комп'ютерними шахраями, можуть тільки найбільші оптимісти. На кожну хитрість, на кожен замок у злодія завжди знайдеться своя відмичка. Тому максимум, що можна зробити для безпечної роботи на ПК, – установити спеціальне програмне забезпечення, яке якщо не зможе запобігти, то хоча б зведе до мінімуму ризик крадіжки даних.

У цьому сенсі *Steganos Privacy Suite 2012* – це відмінний вибір. Програми пакета вже протягом багатьох років надійно оберігають секрети багатьох тисяч користувачів.

Хід роботи

Як засіб приховання даних слід використовувати пакет утиліт приховування та шифрування даних *Steganos Privacy Suite 15*.

Для використання функцій стеганографії у *Steganos Privacy Suite* необхідно зі *Steganos Privacy Main Menu* вибрати пункт "Шифровка & Скрытие" (рис. 5.16, 5.17) або запустити "Шифровка & Скрытие" з головного меню ОС.

Рис. 5.16. **Головне вікно пакета Steganos Security Suite 15**

У вікні "Приховування та шифрування файлів" (див. рис. 5.17) виберіть "Створити зашифрований файл" для створення нового проєкту шифрування та приховування даних.

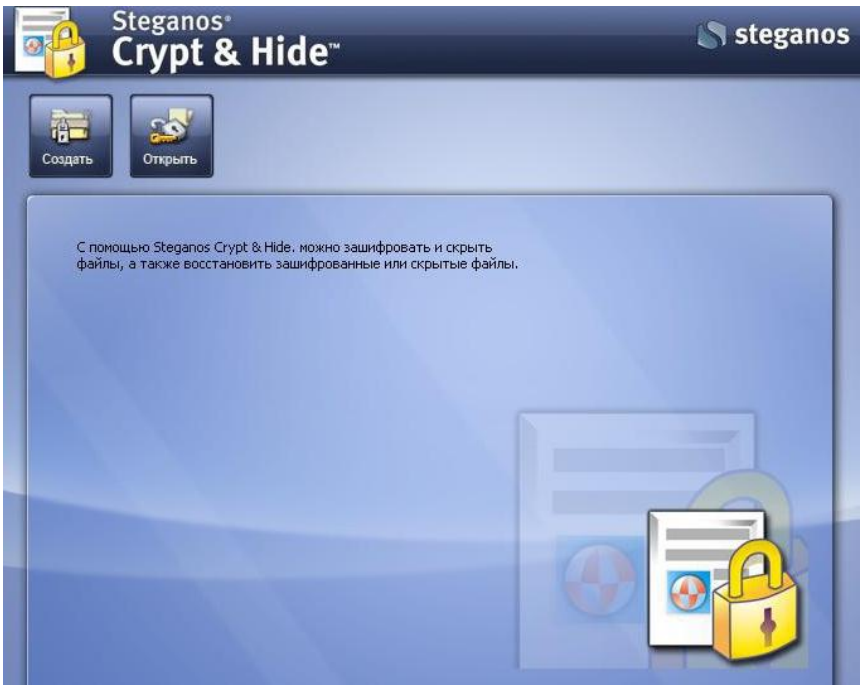


Рис. 5.17. **Головне вікно "Приховування та шифрування файлів"**

У новий проєкт можна додати файли, які необхідно приховати в архіві з розширенням **.edf. Для цього натисніть "Файл" або "Папка".

Для демонстрації можливостей вікна "Приховування та шифрування файлів" додайте у проєкт кілька різнотипних файлів (рис. 5.18).

Рис. 5.18. **Новий проєкт "Приховування та шифрування файлів"**

Після додавання файлів у проєкт вони з'являться в таблиці "Приховування та шифрування файлів" (рис. 5.19 і 5.20).

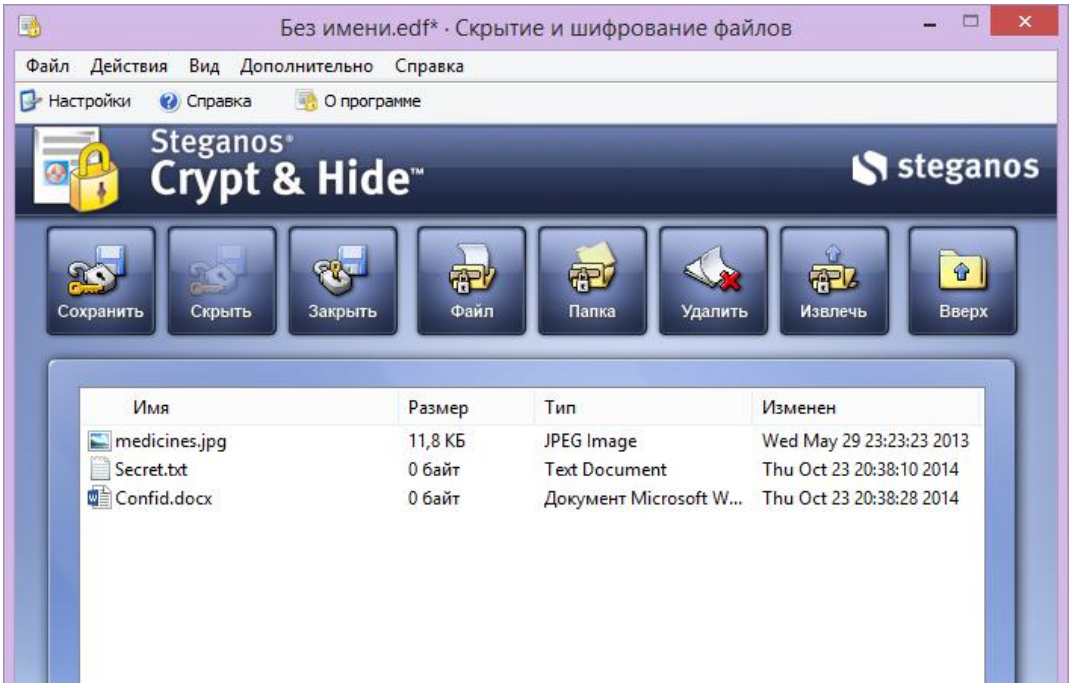


Рис. 5.19. **Проєкт "Приховування та шифрування файлів" із доданими файлами**

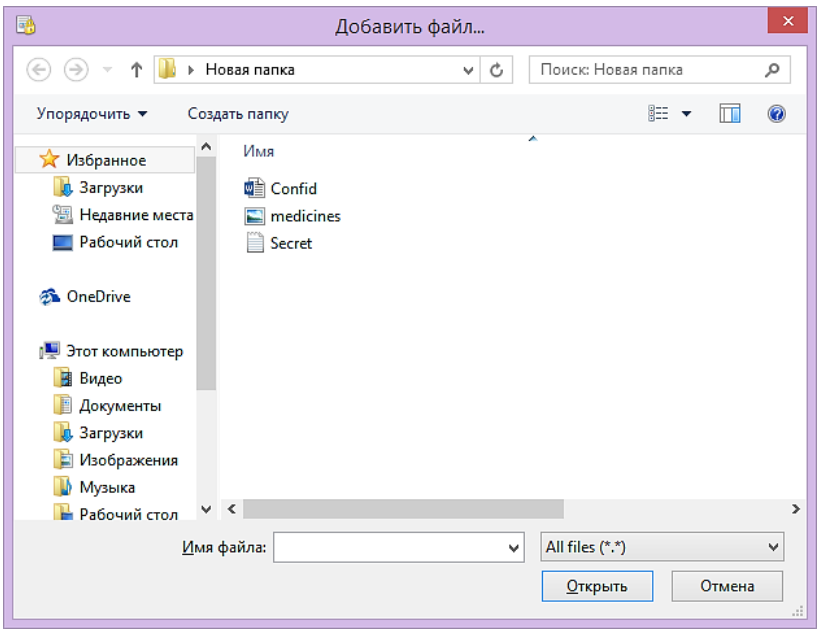


Рис. 5.20. **Додавання файлів у проєкт "Приховування та шифрування файлів"**

Водночас файли, які є звичайним текстом (рис. 5.21), документом WordPad (рис. 5.22) і графічним файлом у форматі JPG (рис. 5.23).

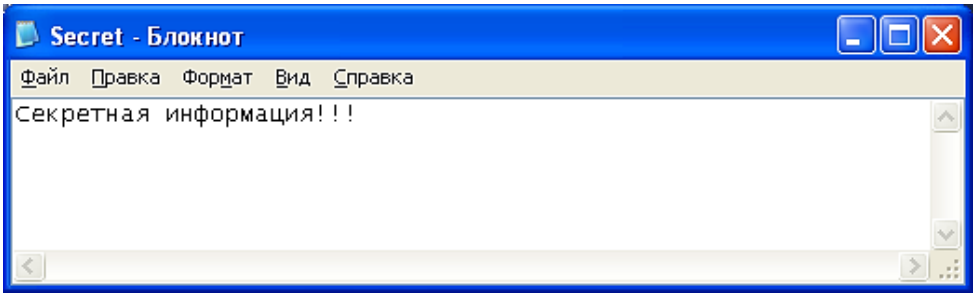


Рис. 5.21. **Уміст приховуваного файлу Secret.txt**

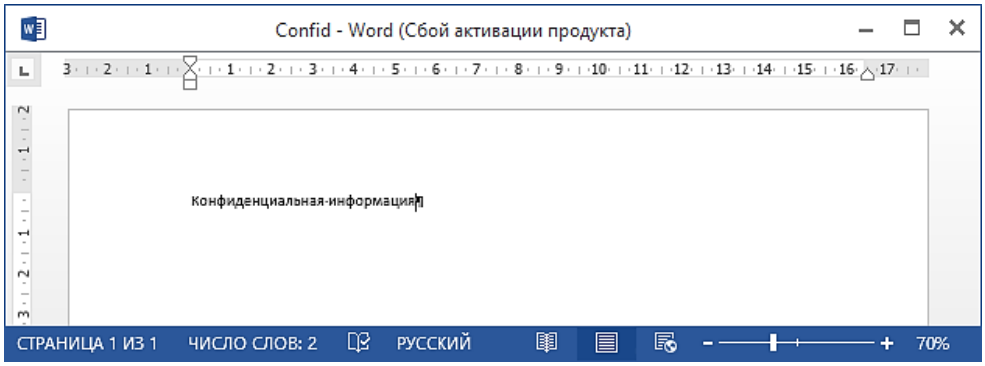


Рис. 5.22. **Уміст приховуваного файлу Confid.doc**

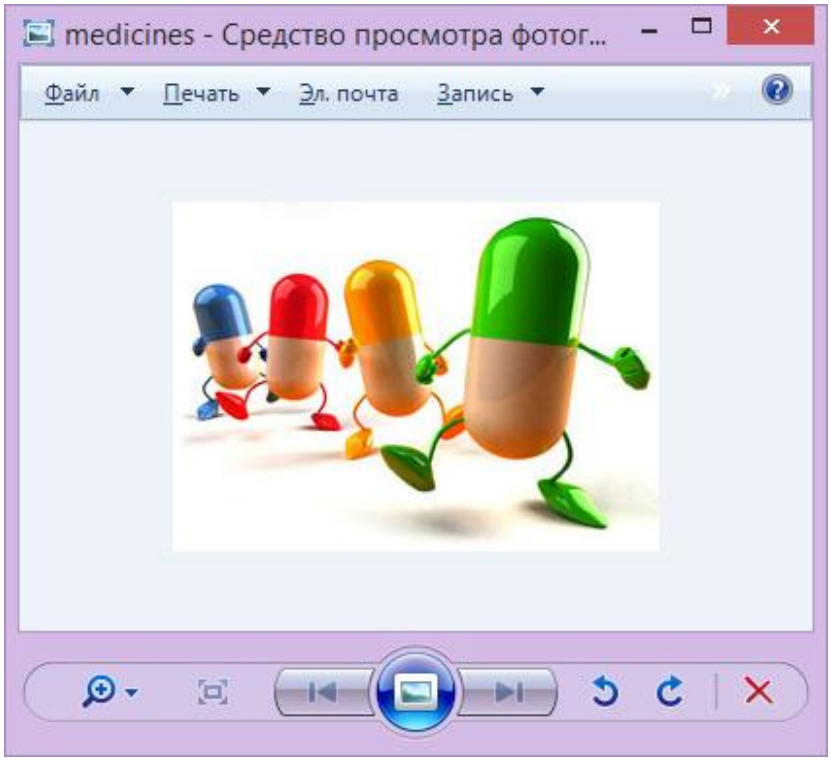


Рис. 5.23. **Уміст приховуваного файлу ZI.jpg**

Після того як сформовано список файлів для приховування, необхідно їх зашифрувати. Для цього потрібно вибрати "Закрыть". Після цього програма пропонує задати пароль для шифрування секретних файлів (рис. 5.24).

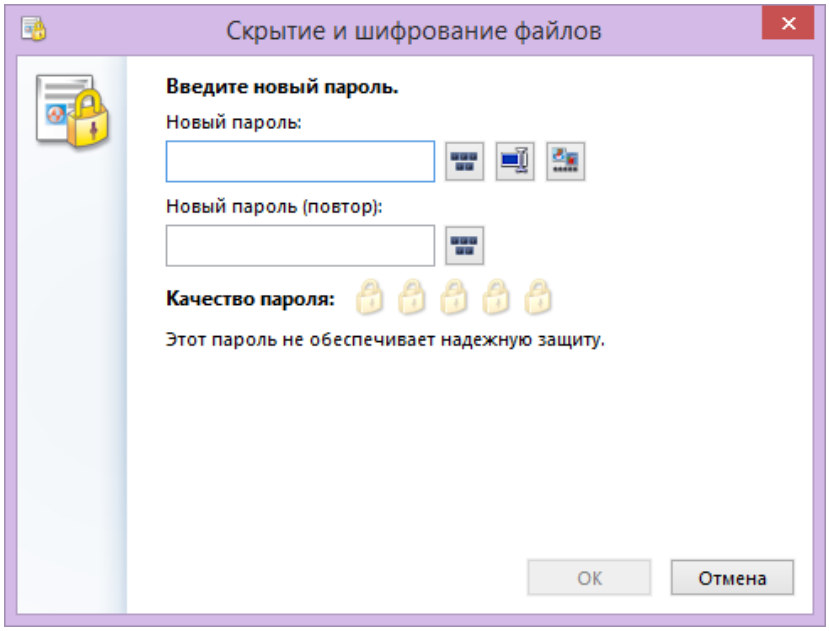


Рис. 5.24. **Установлення пароля**

Таким чином, у користувача є три варіанти встановлення пароля: уведення текстового пароля вручну, генерація пароля за заданими критеріями складності (рис. 5.25) і графічним способом за допомогою утиліти "Пароль-картинка" (рис. 5.26).

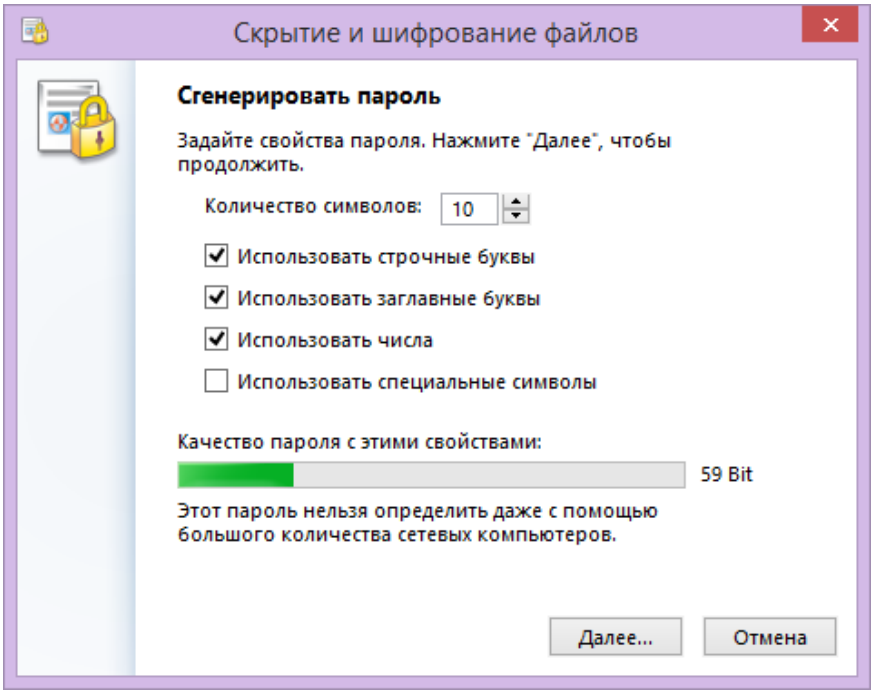


Рис. 5.25. **Генерація пароля**

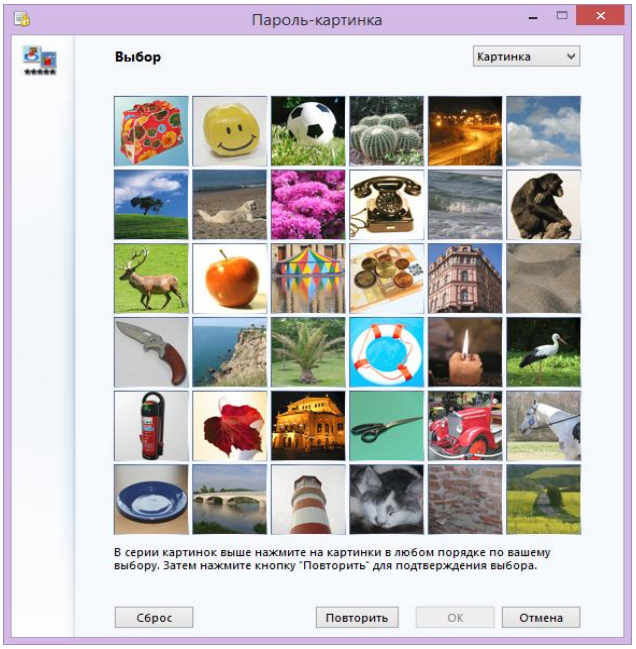


Рис. 5.26. **Установлення пароля за допомогою утиліти "Пароль-картинка"**

Такий вигляд має зашифрований архів із секретними файлами (рис. 5.27).

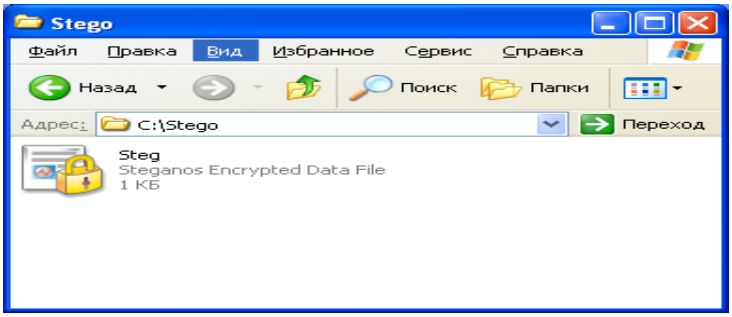


Рис. 5.27. **Зашифрований архів Steg**

Для розшифрування секретного архіву за допомогою "Шифровка & Скрытие" відкрийте ваш архів, натиснувши кнопку "Відкрити", знайдіть зашифрований архів і введіть ваш пароль (рис. 5.28).

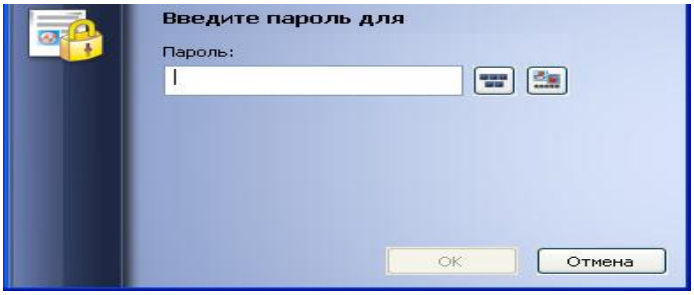


Рис. 5.28. **Уведення пароля для розшифрування архіву із секретними файлами**

Із розшифрованим архівом (рис. 5.29) можна виконати багато файлових операцій. Можна копіювати файли та папки, додавати файли, зберегти зміни, закрити архів, тобто зашифрувати його знову. Також є можливість приховати зашифрований архів у файлах JPG, WAV або BMP.

Рис. 5.29. **Розшифрований архів Steg.edf**

Алгоритм шифрування. Програма підтримує 256-бітове шифрування за стандартом AES. Стандарт AES (Advanced Encryption Standard) є стандартом шифрування США, узятим 2000 року. Він використовує алгоритм Rijndael. Цей алгоритм є симетричним блоковим шифром, який працює із блоками даних довжиною 128 бітів і використовує ключі довжиною 128, 192 і 256 бітів (версії AES-28; AES-192 і AES-256). Сам алгоритм може працювати й з іншими довжинами блоків даних і ключів, але ця можливість до стандарту не ввійшла. В AES основним є поліноміальне подання кодів. Так, байт { 01100011 } слід подати як:

$$x^6 + x^5 + x + 1.$$

Алгоритм AES виконує операції над двовимірними масивами байтів, так званими станами (*state*). Стан складається із 4 рядків по N_b байтів. N_b дорівнює довжині блока, поділеній на 32 (у цьому стандарті $N_b = 4$). Це дозволяє позначати структуру як *sr, c* або *s*[*r, c*], де $0 \le r < 4$ і $0 \le c < 4$. Вхідний код (*in*), який є послідовністю із 16-ти байтів, можна подати як:

$$s[r,c] = in[r + 4c].$$

Для реалізації алгоритму AES використовують операції додавання байтів (за модулем 2 = XOR) і множення. В алгоритмі AES для множення байтів використовують незвідний многочлен:

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

Обчислення добутку *M* байтів {*b1*} на {*b2*} тут виконують, згідно з таким алгоритмом:

$$M = [\{b1\} \times \{b2\}] \ mod \ m(x).$$

У цьому разі обернена величина байта дорівнює:

$$\{b\}^{-1} = \{b\} \ mod \ m(x).$$

Для множення напівбайтів використовують незвідний поліном $m^2(x) = x^4 + 1$. Обчислення добутку M напівбайтів $\{a\}$ на $\{b\}$ тут виконують таким чином:

$$M = [\{a\} \times \{b\}] \; mod \; m2(x),$$

де M – це напівбайт d.

Операцію множення напівбайтів $\{a\}$ на $\{b\}$ можна записати в матричному вигляді:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Відомо, що довжини ключів N_k (довжина, виміряна у 32-бітних словах) можуть набувати значень 4, 6 або 8 (для AES-128, 192 і 256, відповідно). Кількість ітерацій N_r (round), реалізованих в алгоритмі AES, становить, відповідно 10, 12 і 14.

Приховування інформації в зображенні. Зашифрований архів, як і будь-які файли, можна приховати у стегаконтейнер. Для цього необхідно виділити мишкою в зашифрований архів або файли, які треба приховати, і, натиснувши на них правою кнопкою мишки, вибрати пункт Hide (рис. 5.30 і 5.31).

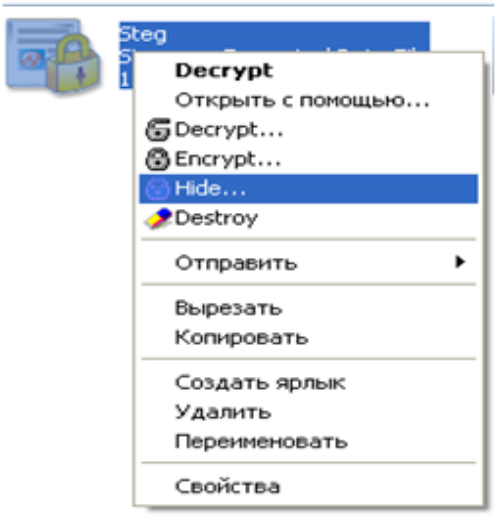


Рис. 5.30. **Приховування зашифрованого архіву**

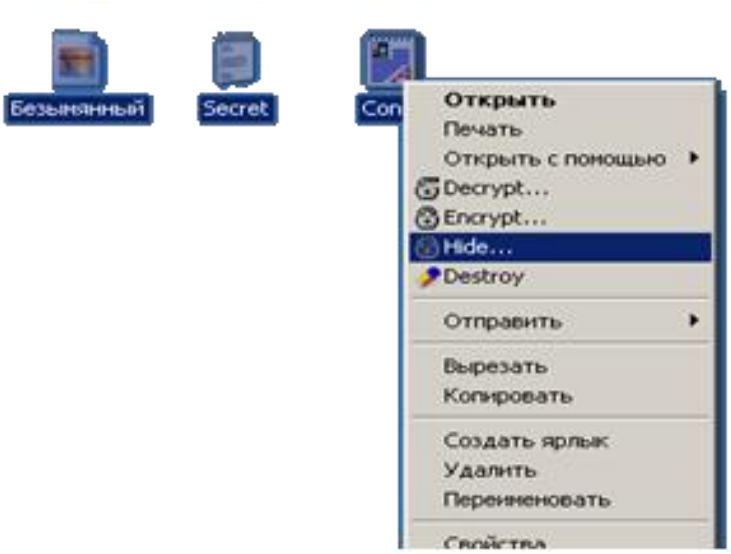


Рис. 5.31. **Приховування файлів**

Далі Steganos File Manager запропонує (рис. 5.32) вибрати вручну стеганоконтейтер (нерухоме графічне зображення), у якому буде прихована інформація. Виберіть контейнер самостійно і як такий використовуйте графічний файл у форматі JPEG (*.jpg, *.JPG), розмір якого перевищує сумарний розмір прихованої інформації.

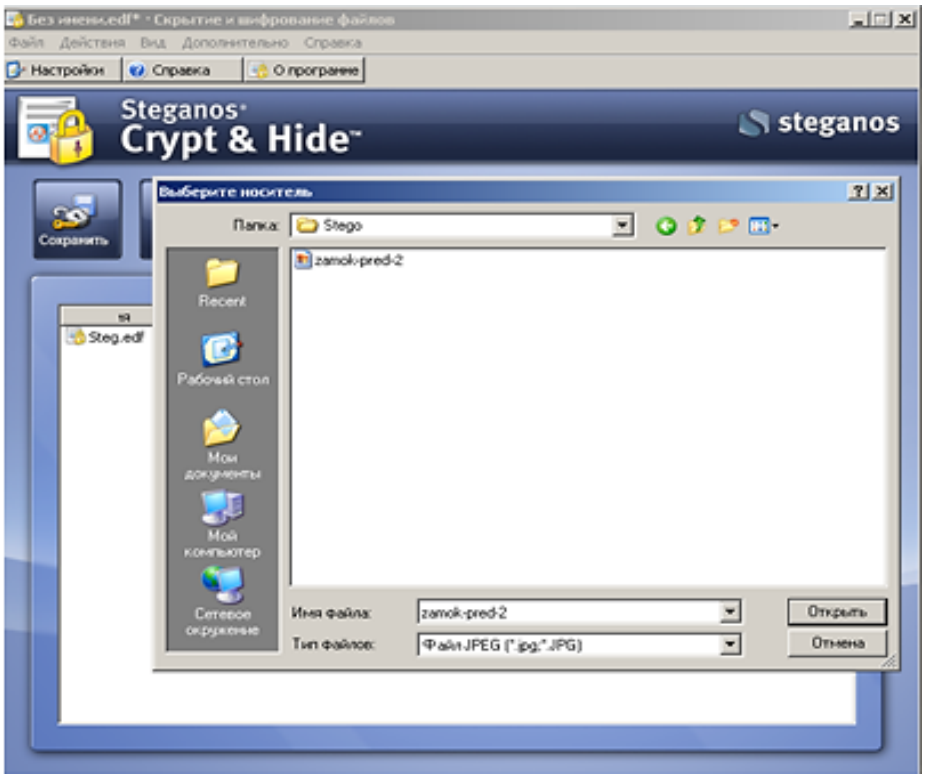


Рис. 5.32. **Діалог вибору контейнера**

Після вибору контейнера Steganos File Manager пропонує задати пароль для шифрування секретних файлів (рис. 5.33).

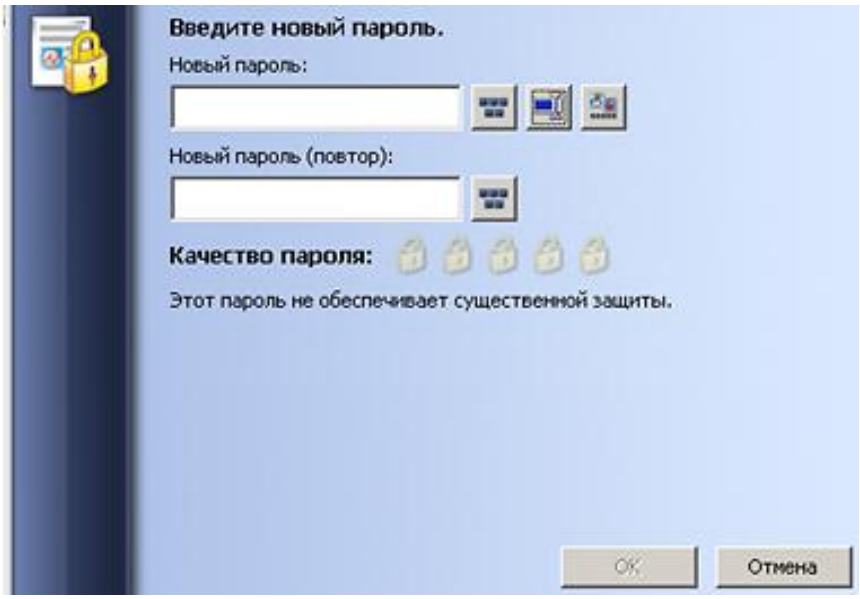


Рис. 5.33. **Уведення нового пароля**

Отже, у користувача є три варіанти встановлення пароля: уведення текстового пароля вручну, генерація пароля за заданими критеріями складності (рис. 5.34) й установлення пароля графічним способом за допомогою утиліти "Пароль-картинка".

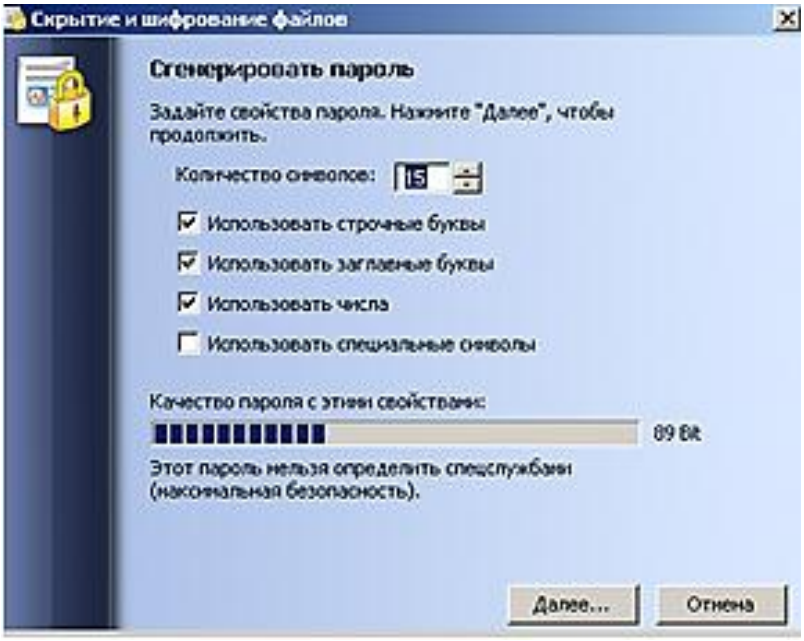


Рис. 5.34. **Генерація пароля**

Після встановлення пароля потрібно натиснути "Далее", щоб виконати шифрування та приховування в контейнері секретних файлів. У результаті в контейнері буде розміщено секретні файли в зашифрованому вигляді (алгоритм шифрування – AES 256 bit), а Steganos File Manager запропонує видалити або залишити незашифровані секретні файли.

Для порівняльного аналізу вихідного контейнера зі стегаконтейнером можна використовувати програму Image Discerner, яка дозволяє зробити спектральний аналіз двох зображень: порівнює файли в різних режимах (використовують 9 кольорових фільтрів) і будує карту відмінностей.

Головне вікно Image Discerner зображено на рис. 5.35.

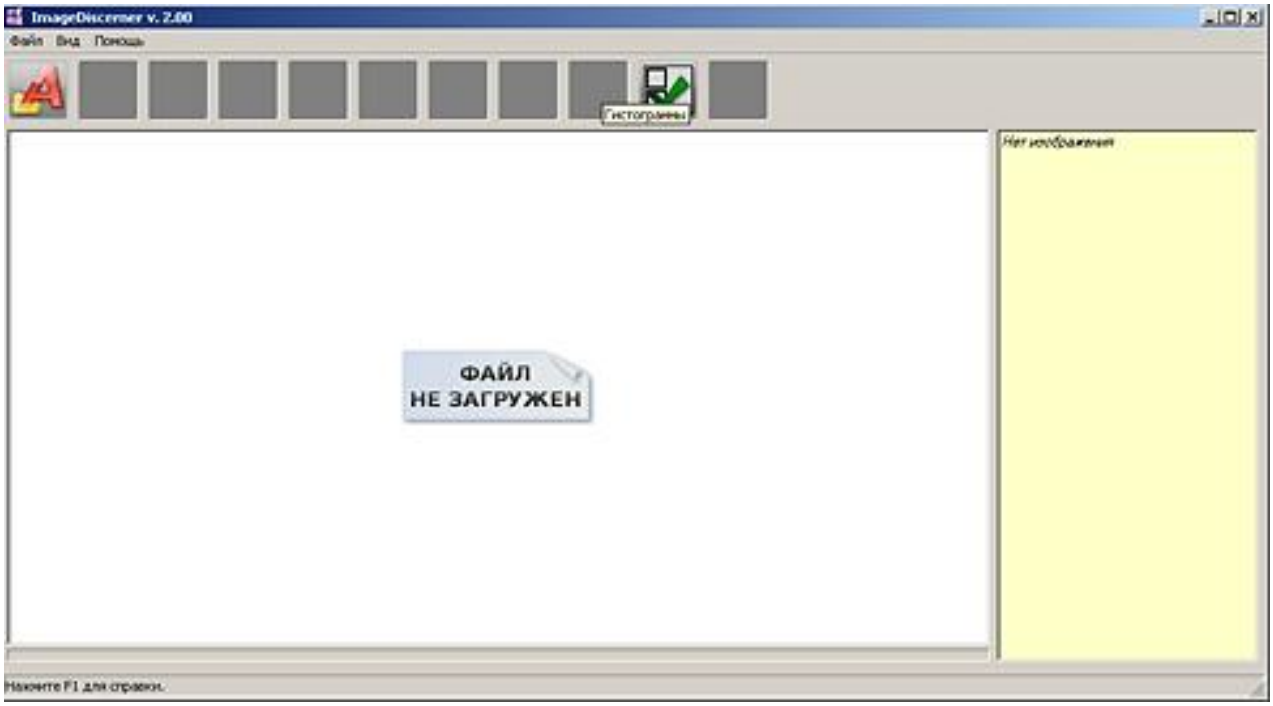


Рис. 5.35. **Головне вікно Image Discerner**

Для порівняння необхідно завантажити два файли зображень: малюнок і стегоконтейнер (малюнок із вбудованою конфіденційною інформацією) (рис. 5.36, 5.37), натиснути кнопку "Сравнить изображения" на панелі інструментів, налаштувати режим порівняння, для чого потрібно буде вибрати картку відмінностей і ступінь контрасту (рис. 5.38).

Оскільки у Steganos Privacy Suite приховування інформації досягають шляхом зміни кожного LSB (least significant bit – молодший біт), то для чіткого візуального виявлення відмінностей рекомендовано порівняння контейнерів виконувати зі збільшенням у 2 550 разів.

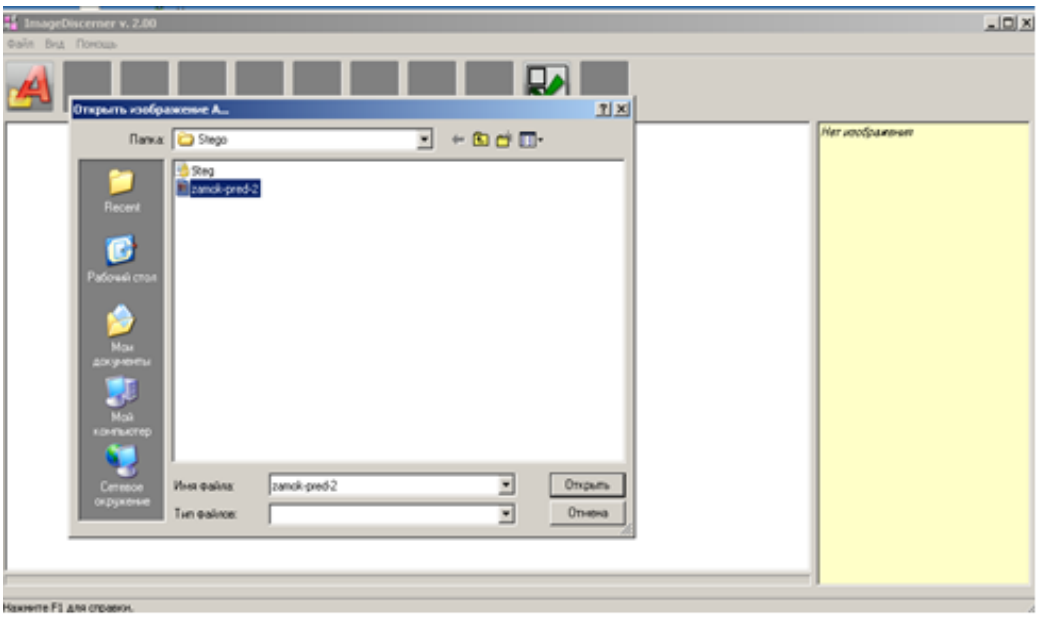


Рис. 5.36. **Відкриття зображення A**

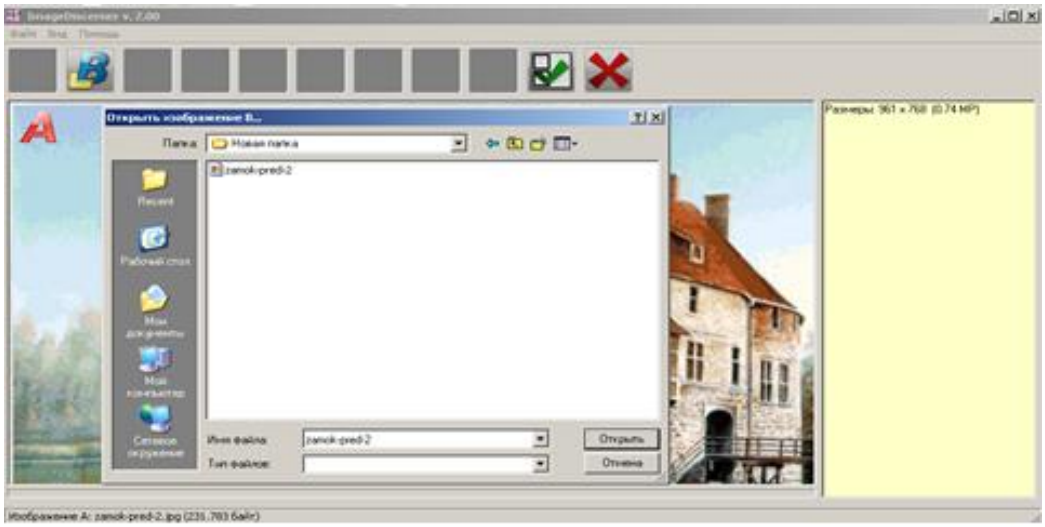


Рис. 5.37. **Відкриття зображення B**

Рис. 5.38. **Порівняння зображень**

Якщо у процесі порівняння виберуть, наприклад режим HUE (відтінок), то дістануть карту відмінностей (рис. 5.39) і для візуального порівняння – вихідне зображення (рис. 5.40). На цій карті видно, що секретну інформацію приховано в найконтрастніших областях зображення.

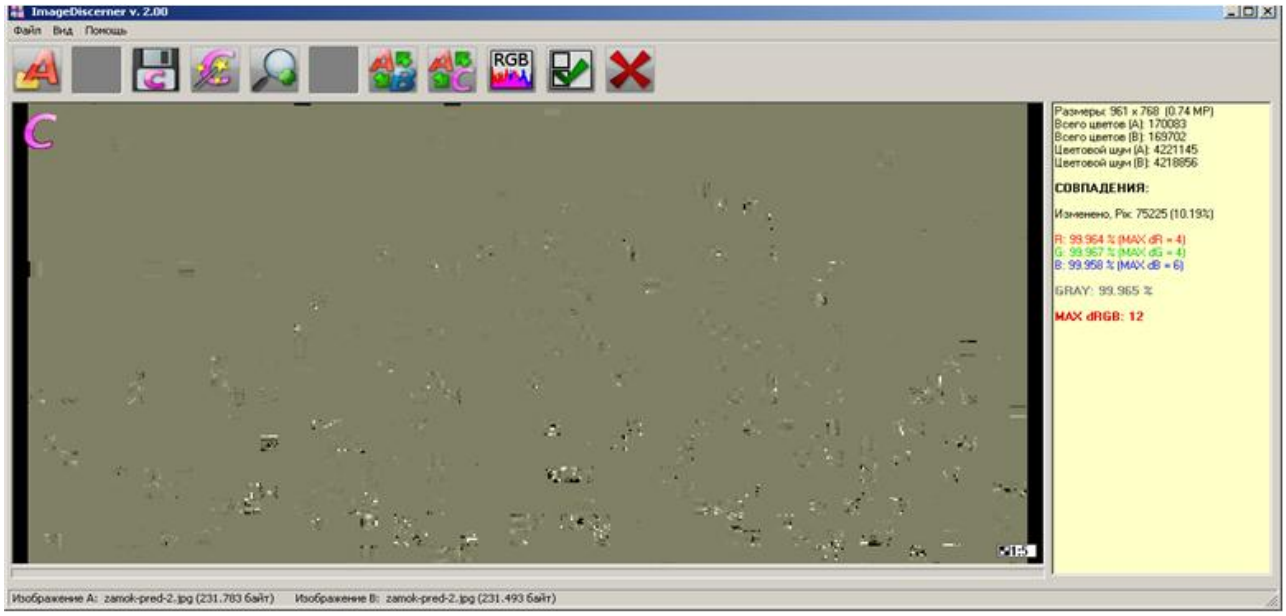


Рис. 5.39. **Карта відмінностей у режимі HUE**

Рис. 5.40. **Початкове зображення**

5.4. Завдання до лабораторної роботи

1. За допомогою описаного програмного забезпечення виконайте приховування конфіденційної інформації (ПІБ студента) різними методами, оцініть їхні переваги й недоліки. Кількість речень, тобто кількість входжень у текст "." (крапка – пробіл) (COUNT) має такі обмеження: 0 < COUNT ≤ 36 120, що означає, що можна приховати до 36 120 символів. Рекомендовано використовувати COUNT < 5000 для найбільш ефективної роботи.

2. За допомогою пакета програмного продукту і пакета Steganos Privacy Suite зашифруйте та приховайте у графічному контейнері декілька файлів. Порівняйте порівняння вихідного контейнера зі стегоконтейнером за основними параметрами файлової системи (розмір файлу, дата й час створення та модифікації файлу, додаткові атрибути тощо), зробіть висновки. Проаналізуйте можливе співвідношення розміру контейнера із прихованою інформацією. За допомогою програми Image Discerner зробіть порівняння на основі спектрального аналізу вихідного графічного файлу зі стегоконтейнером у різних режимах (наприклад, у різних каналах кольорів).

3. Підготуйте звіт установленого зразка, у висновках сформулюйте результати теоретичних досліджень і практичних умінь із використанням різних способів приховування інформації на основі цифрової стеганографії.

5.5. Контрольні запитання

1. Що таке "стеганографія"?
2. Чим відрізняється класична стеганографія від цифрової?
3. Де застосовують методи цифрової стеганографії?
4. Що може бути контейнером у стегосистемі?
5. Які критерії характеризують якість стегосистеми?

Лабораторна робота 6
Використання програми PGP для шифрування повідомлень електронної пошти

6.1. Мета

Мета цієї лабораторної роботи полягає в набутті студентами практичних навичок у роботі із програмними засобами забезпечення безпеки електронної пошти за допомогою шифрування та цифрового підпису.

6.2. Рекомендації до підготовки до виконання

Для підготовки до лабораторної роботи необхідно вивчити основні поняття та визначення з питань PGP; ознайомитися з описом лабораторної роботи та її програмним забезпеченням; підготувати бланк звіту, згідно з розділом "Зміст звіту"; підготувати відповіді на контрольні запитання.

6.3. Загальні теоретичні положення

Короткий опис системи PGP. Систему PGP (Pretty Good Privacy – англ. "цілком безпечна приватність") засновано на алгоритмах, які витримали перевірку практикою, їх уважають винятково надійними. Зокрема, до пакета додано алгоритми шифрування з публічним ключем RSA, DSS, алгоритм Діффі – Геллмана, алгоритми традиційного шифрування CAST-128, IDEA і 3DES, а також алгоритм гешування SHA-1.

Система PGP має широку сферу застосування: від корпорацій, які хочуть мати стандартизовану схему шифрування файлів і повідомлень, до пересічних користувачів, які потребують захисту свого листування з іншими користувачами в інтернеті або якійсь іншій мережі. Систему PGP не було розроблено урядовою або будь-якою іншою офіційною організацією, і тому вона не контролюється ними. Отже, PGP має додаткову привабливість для людей з інстинктивною недовірою до "апарату".

Спочатку слід розглянути загальні принципи роботи PGP, потім з'ясувати, як створюють та зберігають криптографічні ключі, і, нарешті,

обговорити важливе питання управління публічними ключами. Перелік позначень, використовуваних у подальшому для розгляду матеріалу, такі:

K_a – сеансовий ключ, використовуваний в схемі традиційного шифрування;

KR_a – приватний ключ А, використовуваний у схемі шифрування з відкритим ключем;

KU_a – відкритий ключ А, використовуваний у схемі шифрування з відкритим ключем;

EP – шифрування у схемі з відкритим ключем;

DP – дешифрування у схемі з відкритим ключем;

EC – шифрування у схемі традиційного шифрування;

DC – дешифрування у схемі традиційного шифрування;

H – функція гешування;

|| – конкатенація;

Z – стиснення за допомогою алгоритму ZIP;

R64 – перетворення на формат radix-64 ASCII.

Сервіс PGP, крім управління ключами, складається із п'яти функцій: автентифікації, конфіденційності, стискання, сумісності на рівні електронної пошти та сегментації (табл. 6.1).

Таблиця 6.1

Стисла характеристика функцій системи PGP

Функції	Алгоритми	Опис
1	2	3
Цифровий підпис	DSS/SHA або RSA/SHA	За допомогою SHA-1 створюють геш-код повідомлення. Визначений у такий спосіб профіль повідомлення шифрують за допомогою DSS або RSA з використанням приватного ключа відправника та додають у повідомлення
Шифрування повідомлення	CAST, IDEA, "потрійний" DES із трьома ключами й алгоритмом Діффі – Геллмана або RSA	Повідомлення шифрують за допомогою CAST-128, IDEA, 3DES з одноразовим сеансовим ключем, згенерованим відправником. Сеансовий ключ шифрують за допомогою алгоритму Діффі – Геллмана або RSA з використанням відкритого ключа отримувача та додають у повідомлення

Закінчення табл. 6.1

1	2	3
Стискання	ZIP	Повідомлення можна стиснути для зберігання або передавання, використовуючи ZIP
Сумісність на рівні електронної пошти	Перетворення на формат radix-64	Щоб забезпечити прозорість для всіх застосунків електронної пошти, шифроване повідомлення можна перетворити на рядок ASCII, використовуючи перетворення на формат radix-64
Сегментація	–	Щоб задовольнити обмеженням максимального розміру повідомлень, PGP виконує сегментацію та зворотне складання повідомлення

На рис. 6.1а показано схему сервісу цифрового підпису, яку пропонує PGP. Тут виконують таку послідовність дій. Відправник створює геш-образ повідомлення. Для цього використовує алгоритм SHA-1, який забезпечує 160-бітовий геш-код. Визначений геш-образ шифрують за допомогою алгоритму RSA з використанням приватного ключа відправника і результат додають до повідомлення. Отримувач використовує публічний ключ відправника, щоб дешифрувати та відновити геш-образ. Отримувач обчислює новий геш-образ отриманого повідомлення та порівнює його з дешифрованим. Якщо вони збігаються, повідомлення вважають справжнім.

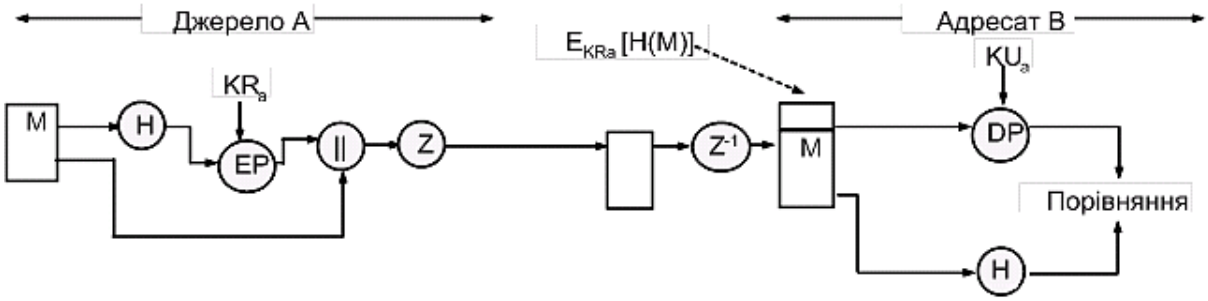
Комбінація SHA-1 і RSA забезпечує ефективну схему цифрового підпису. Зважаючи на надійність RSA, отримувач упевнений у тому, що тільки власник відповідного приватного ключа міг створити цей підпис. Надійність SHA-1 дає йому впевненість, що третя особа не могла створити інше повідомлення з відповідним геш-кодом і, отже, із підписом оригінального повідомлення. Підписи можна генерувати з допомогою DSS/SHA-1.

Хоча підписи зазвичай приєднують до повідомлень або файлів, для яких їх створюють, процес не завжди відбувається саме так, підтримують відокремлені підписи. Відокремлений підпис можна зберігати та передавати окремо від самого повідомлення. Це є корисним у багатьох випадків. Наприклад, відокремлений підпис програмного продукту може згодом допомогти у виявленні зараження програми вірусом, користувачі можуть мати окремі протоколи для підписів та оригінальних повідомлень. Нарешті,

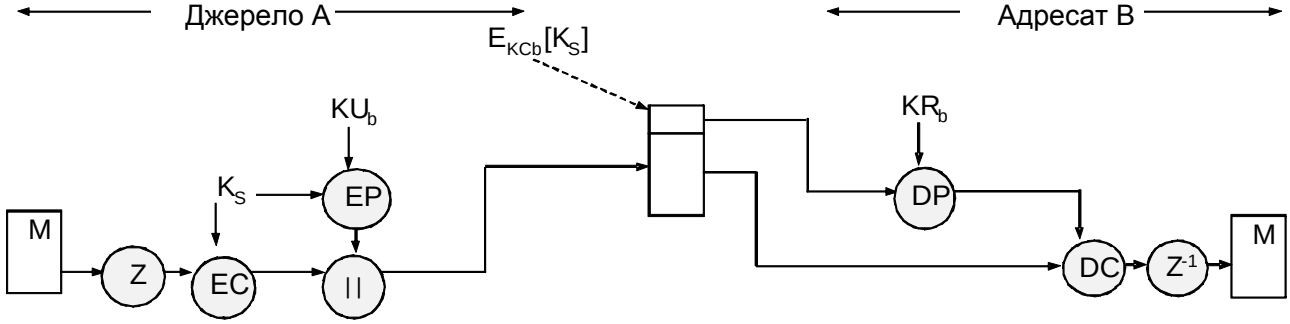
такі підписи можна використовувати тоді, коли підписувати документ має не одна, а більше сторін, як, наприклад, у разі контракту. Підпис кожної зі сторін тоді є незалежним і, таким чином, його застосовують тільки до цього документа. Інакше підписи мають бути вкладеними, щоб третя сторона підписувала документ разом із підписом першої сторони тощо.

Для забезпечення конфіденційності та сервісу автентифікації на прикладному рівні в комп'ютерних системах і мережах використовують схеми PGP і S/MIME (Secure/Multipurpose Internet Mail Extension).

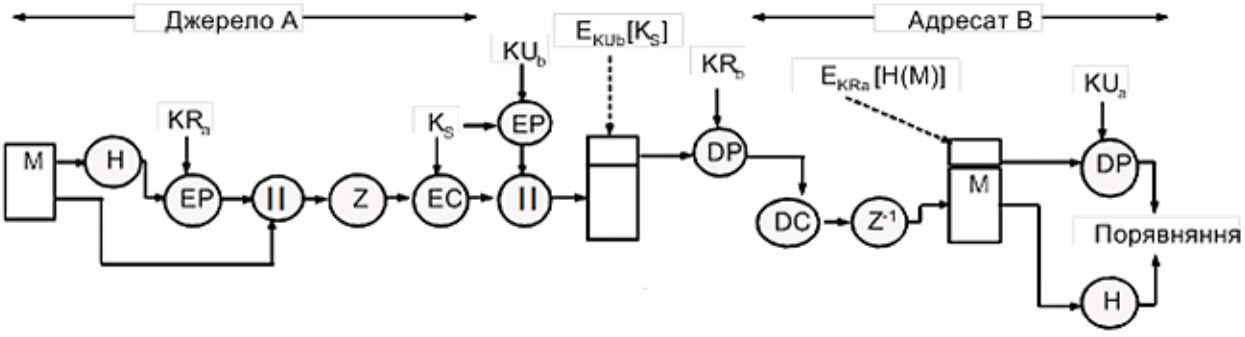
На рис. 6.1 показано основні схеми використання системи PGP.



а



б



в

Рис. 6.1. **Схеми використання системи PGP: а – тільки автентифікація; б – тільки конфіденційність; в – конфіденційність та автентифікація**

Іншим основним сервісом PGP є конфіденційність, яку забезпечують шифруванням повідомлень, призначених для передавання або зберігання у вигляді файлів. В обох випадках можна використовувати традиційне шифрування за допомогою алгоритму CAST-128, IDEA або 3DES. Алгоритми можна використовувати в режимі зворотного зв'язку за шифром (режим CFB).

Як завжди, необхідно вирішувати проблему розподілу ключів. У PGP кожний ключ схеми традиційного шифрування застосовують тільки один раз. Це означає, що для кожного повідомлення генерують новий ключ у вигляді випадкового 128-бітового числа. Таким чином, хоча в документації такий ключ називають сеансовим, насправді, він є одноразовим. Через те що ключ використовують одноразово, такий сеансовий ключ приєднують до повідомлення та передають разом із ним. Щоб захистити ключ, його шифрують на публічному ключі отримувача.

На рис. 6.1б показано відповідну схему, яку може бути описано в такий спосіб:

- відправник генерує повідомлення та випадкове 128-бітове число, яке буде сеансовим ключем тільки для цього повідомлення;
- повідомлення шифрують за допомогою алгоритму CAST-128 (або IDEA, або 3DES) на цьому сеансовому ключі;
- своєю чергою, сеансовий ключ шифрують за допомогою алгоритму *RSA* на публічному ключі отримувача та приєднують до початку повідомлення;
- отримувач використовує RSA із приватним ключем, щоб розшифрувати й тим самим відновити сеансовий ключ;
- сеансовий ключ застосовують для розшифрування повідомлення.

Щоб забезпечити альтернативу використанню RSA для шифрування ключа, у PGP пропонують алгоритм погодження ключа Діффі – Гелльмана. Насправді, у PGP використовують варіант цього алгоритму з можливостями шифрування/розшифрування, відомий як алгоритм Ель-Гамаля.

У зв'язку із цим слід зробити кілька зауважень. По-перше, щоб зменшити час шифрування, перевагу надають використанню комбінації традиційного шифрування та шифрування з публічним ключем, а не простому використанню RSA або алгоритму Ель-Гамаля, коли повідомлення шифрують безпосередньо ними: CAST-128 та інші алгоритми традиційної схеми шифрування значно швидші. По-друге, використання алгоритмів шифрування з публічним ключем вирішує проблему розподілу сеансових ключів, тому що тільки отримувач має можливість відновити

сеансовий ключ, приєднаний до повідомлення. У цій ситуації радше кожне повідомлення є незалежною одиночною подією зі своїм власним ключем. До того ж унаслідок самої природи електронної пошти, яка є системою із проміжним зберіганням даних, використання процедури підтвердження зв'язку, для того щоб переконатися в ідентичності сеансового ключа обох сторін, досить непрактичне. Нарешті, використання одноразових ключів у традиційній схемі шифрування підсилює й без того досить надійний алгоритм традиційного шифрування. Тільки невеликий обсяг відкритого тексту шифрують із використанням одного ключа, і між ключами немає зв'язку. Таким чином, захищеність запропонованої схеми буде визначено стійкістю алгоритму шифрування з публічним ключем. Тому PGP пропонує користувачу вибір для довжини ключа від 768 до 3 072 бітів (довжину ключа DSS для підписів обмежено 1 024 бітами).

Як показано на рис. 6.1в, для одного повідомлення можна використовувати обидві операції. Спочатку для повідомлення генерують підпис, який додають у початок повідомлення. Потім відкритий текст і підпис шифрують за допомогою алгоритму CAST-128 (IDEA або 3DES), а сеансовий ключ шифрують за допомогою RSA (або алгоритму Ель-Гамаля). Така схема зручніша від альтернативної, де підпис генерують уже для зашифрованого повідомлення, оскільки в першому випадку підпис зберігають разом із відкритим текстом. До того ж у разі тристоронньої верифікації, третій стороні не потрібно розшифровувати повідомлення для перевірки підпису.

Отже, із використанням обох служб відправник спочатку підписує повідомлення за допомогою приватного ключа, потім шифрує повідомлення за допомогою сеансового ключа та, нарешті, шифрує сеансовий ключ за допомогою публічного ключа отримувача.

За замовчуванням PGP стискає повідомлення після створення підпису перед шифруванням. Це має сенс із погляду зменшення обсягу даних як для передавання електронної пошти, так і зберігання у вигляді файлів. Дуже важливим є вибір місця застосування алгоритму стискання, позначеного на рис. 6.1 як Z у стискуванні і як Z^{-1} у розпакуванні даних.

Підпис генерують до стискання за такими ознаками:

1) краще підписувати нестиснене повідомлення, щоб у подальшому мати можливість зберігати його у відкритому вигляді разом із підписом. Якщо підписати стиснений документ, то для верифікації необхідно буде зберігати стиснену версію повідомлення або стискати повідомлення щоразу, коли потрібна верифікація. Крім того, такий підхід має додаткові

проблеми через недермінований алгоритм стискання: різні його реалізації виконують стискання в різний спосіб для кращого співвідношення між швидкістю виконання та степенем стискування, отже стиснені повідомлення можуть відрізнятися. Застосування операцій гешування та підпису до стисненого повідомлення змусило б в усіх реалізаціях PGP застосовувати однаковий алгоритм стискання;

2) шифрування повідомлення застосовують після стискання ще й для того, щоб підсилити криптографічний захист повідомлення. Через те що стиснене повідомлення має меншу надмірність мови, порівняно з відкритим текстом, криптоаналіз такого повідомлення значно складніший.

Якщо непотрібно захищати конфіденційність повідомлення, то система шифрує лише геш-образ повідомлення з використанням приватного ключа відправника. Якщо ж задіяний сервіс конфіденційності, то шифрують повідомлення (із використанням одноразового ключа симетричного криптоалгоритму) й електронний цифровий підпис (якщо він є). Однак більшість поштових клієнтів дозволяє використовувати блоки, які містять лише символи ASCII. Для того щоб задовольнити такі вимоги, PGP має сервіс конвертування 8-бітового двійкового потоку в потік ASCII-символів. Для цього використовують схему radix-64. Згідно з нею кожна трибайтова група двійкових даних перетворюється на чотири ASCII-символи, до яких додають контрольну суму (CRC), що дозволяє виявити помилки в передаванні даних.

Конвертування у формат radix-64 збільшує довжину переданого повідомлення на 33 %. Сеансовий ключ і електронний підпис повідомлення відносно компактні, а відкритий текст повідомлення стискається. Фактично надлишкове стискання компенсує розширення, досягнуте, унаслідок перетворення на формат radix-64. Якщо ігнорувати відносно невеликі підпис і ключ, то загальна довжина повідомлення після оброблення буде становити $1{,}33 \times 0{,}5 \times X = 0{,}665 \times X$, де X – довжина відкритого повідомлення. Отже, досягають загального стискання повідомлення приблизно на одну третину.

Додатковим результатом застосування алгоритму radix-64 є те, що він конвертує весь вхідний потік у формат radix-64, навіть якщо вхідне повідомлення вже є символами ASCII. У результаті незашифроване, але конвертоване у формат radix-64 повідомлення буде абсолютно незрозумілим випадковому спостерігачеві, що значно підвищує рівень конфіденційності. PGP можна сконфігурувати так, щоб конвертування у формат radix-64 виконували тільки для підпису відкритого повідомлення. Тоді

отримувач зможе прочитати повідомлення без використання PGP, але його доведеться використовувати, якщо необхідно перевірити підпис.

На рис. 6.2 показано зв'язок між чотирма описаними раніше службами. Під час передавання, якщо це потрібно, підпис генерують за допомогою геш-коду відкритого тексту. Потім відкритий текст і підпис, якщо він є, стискаються.

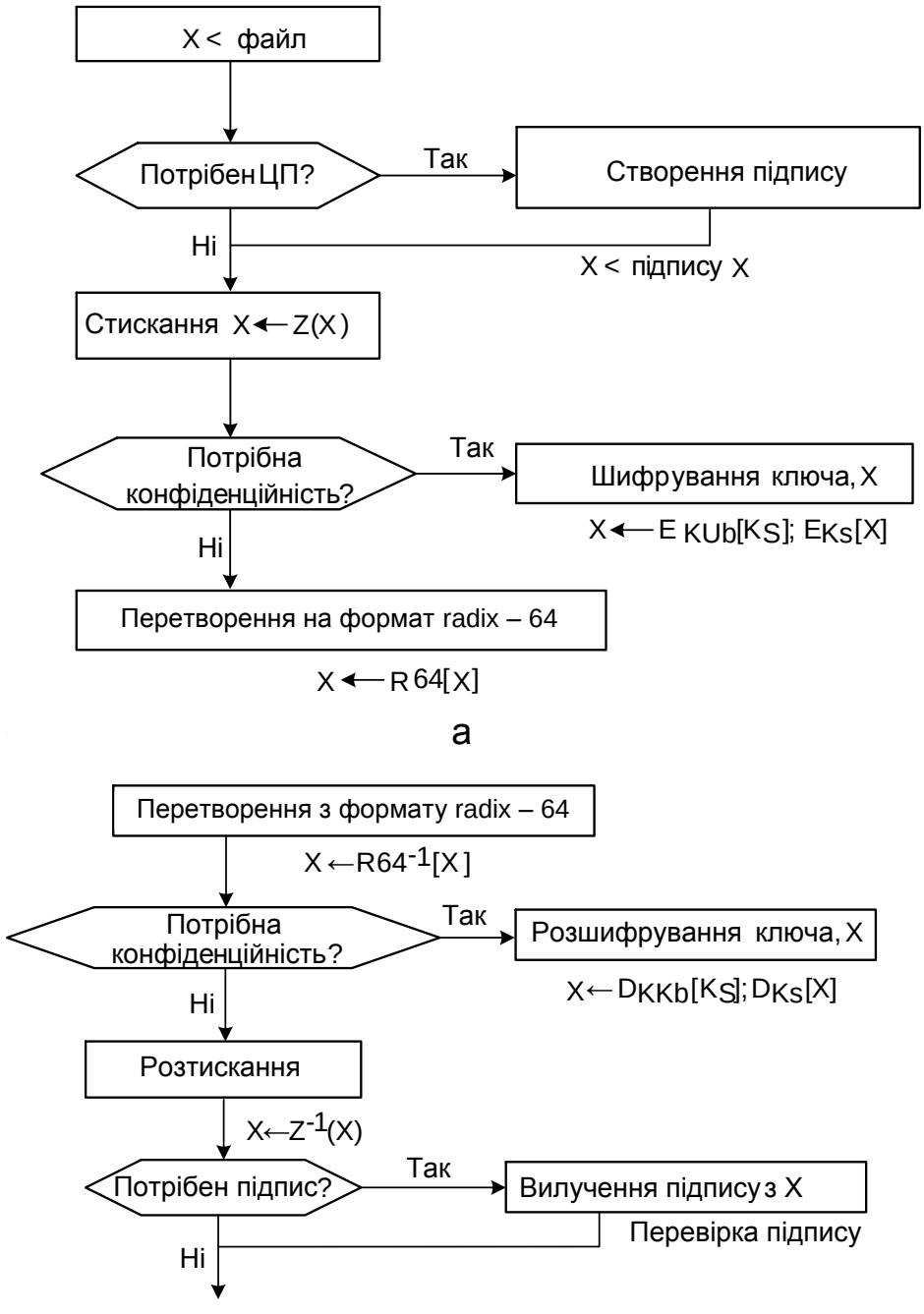


Рис. 6.2. **Відправлення та приймання повідомлень PGP:**
а – загальна схема відправлення (на боці А);
б – загальна схема отримання (на боці В)

Далі, якщо захищають конфіденційність, блок (стиснений відкритий текст або підпис і відкритий текст) шифрують та до його початку додають ключ симетричного алгоритму, зашифрований на публічному ключі отримувача. Нарешті, весь отриманий блок конвертують у формат radix-64.

На боці отримувача вхідний блок спочатку конвертують із формату radix-64 у двійковий. Потім, якщо повідомлення зашифровано, отримувач відновлює сеансовий ключ і розшифровує повідомлення.

Визначений у результаті блок розтискують. Якщо повідомлення підписано, отримувач відновлює отриманий геш-код і порівнює його з гешкодом, обчисленим самостійно.

Засоби електронної пошти часто обмежують максимально допустиму довжину повідомлення. Наприклад, багато засобів електронної пошти, доступних через інтернет, дозволяють пересилання повідомлень довжиною не більшою за 50 000 байтів. Довші повідомлення мають розподіляти на частини, які пересилають окремо. Такий сервіс також убудовано до PGP: довге повідомлення автоматично розподіляють на такі сегменти, які дозволяє пересилати той чи той поштовий клієнт. Сегментацію виконують після всіх інших операцій, додаючи перетворення на формат radix-64.

На боці отримувача система PGP має відкинути заголовок електронної пошти та знову зібрати весь оригінальний блок повідомлення перед виконанням кроків, показаних на рис. 6.2б.

6.4. Завдання для лабораторної роботи

6.4.1. Підготовка до роботи

Вивчити теоретичний матеріал до цієї лабораторної роботи: призначення й основні можливості програми Pretty Good Privacy (PGP) із забезпечення безпеки електронної пошти.

Усвідомити порядок застосування публічного та приватного ключів для захисту електронного листування й гарантії вірогідності джерела повідомлення.

6.4.2. Завдання з експериментальної частини

1. Згенеруйте пари ключів: публічний та приватний.

2. Складіть, зашифруйте й обміняйте шифрованими електронними повідомленнями; розшифруйте отримані повідомлення.

3. Складіть, підпишіть та обміняйтеся електронними повідомленнями; перевірте достовірність джерел повідомлень.

4. Складіть, підпишіть, зашифруйте й обміняйтеся електронними повідомленнями; розшифруйте отримані повідомлення та перевірте достовірність джерел повідомлень.

6.5. Порядок роботи із PGP

Запустіть PGP Desktop, створіть нові ключі,

Для цього необхідно обрати пункт меню File->New PGP Key (рис. 6.3).

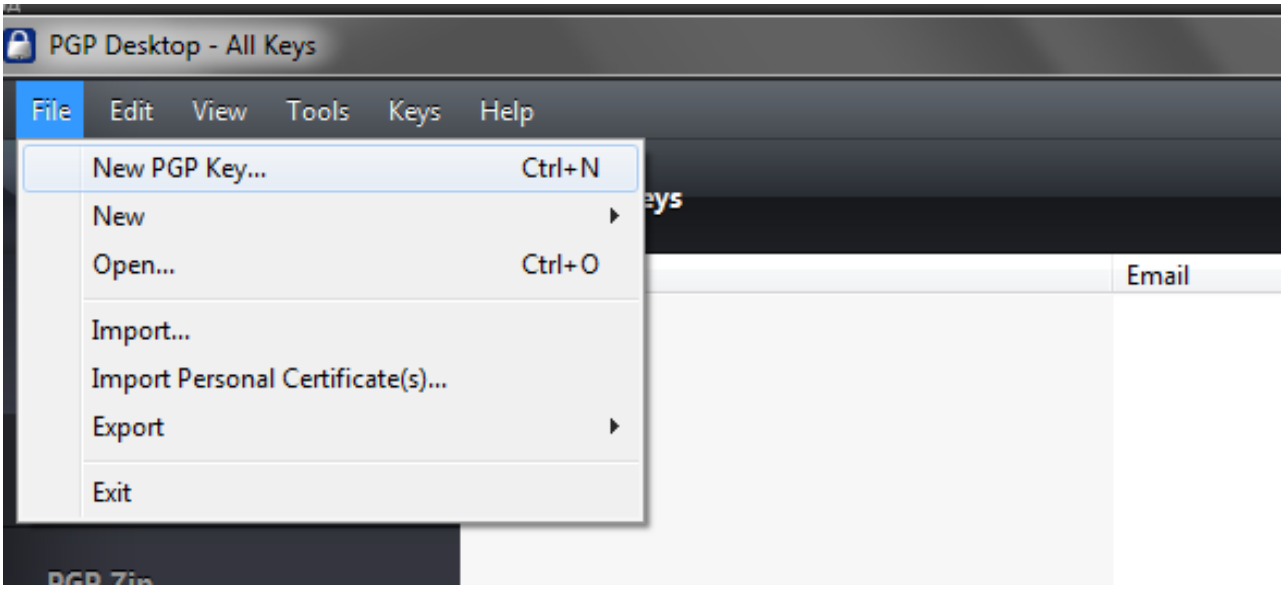


Рис. 6.3. **Створення ключа**

Під час створення ключа вибирають ім'я ключа й адресу електронної пошти (рис. 6.4).

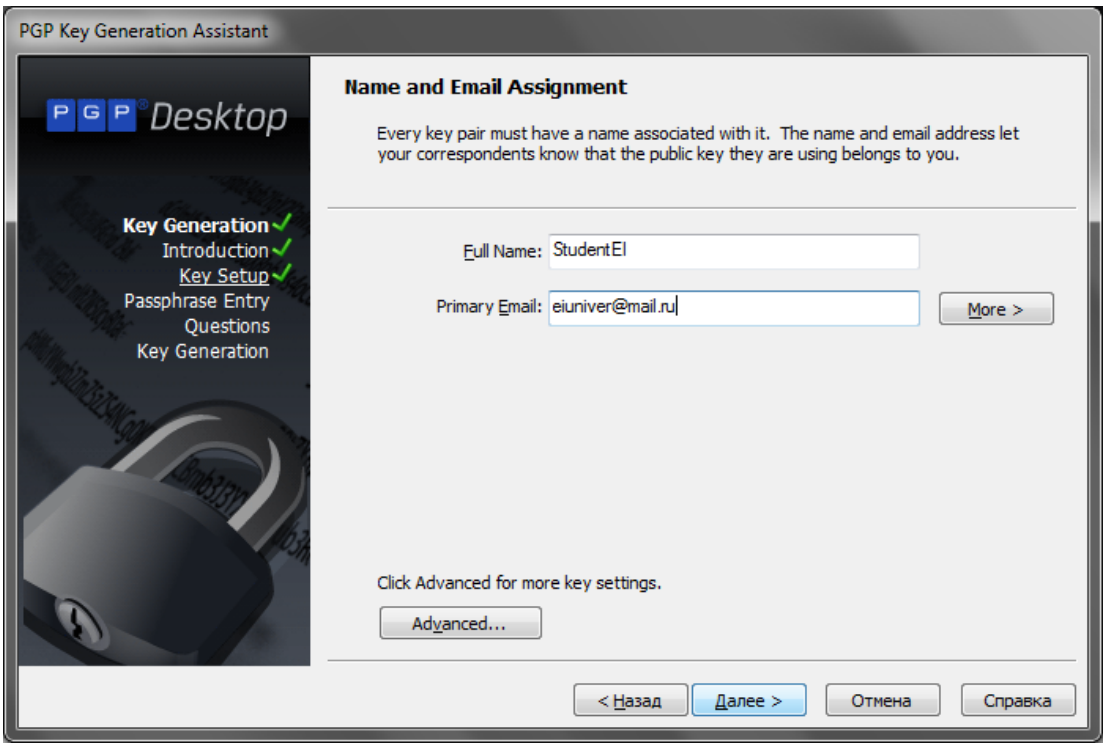


Рис. 6.4. У**ведення параметрів ключа**

Після натискання кнопки Advanced відкривається вікно вибору параметрів ключа (рис. 6.5).

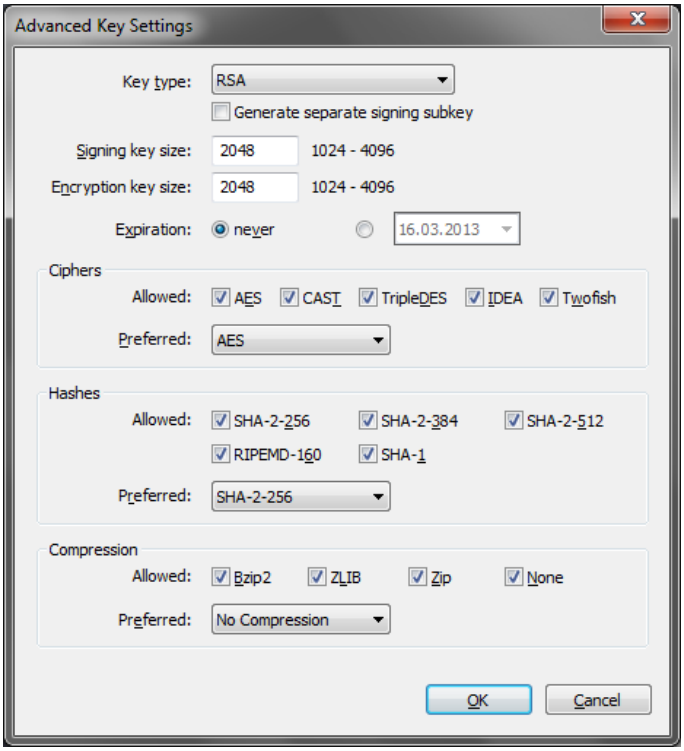


Рис. 6.5. **Вибір параметрів ключа**

Далі система зажадає ввести "фразу-пароль", за допомогою якої можна буде використовувати ключ (рис. 6.6).

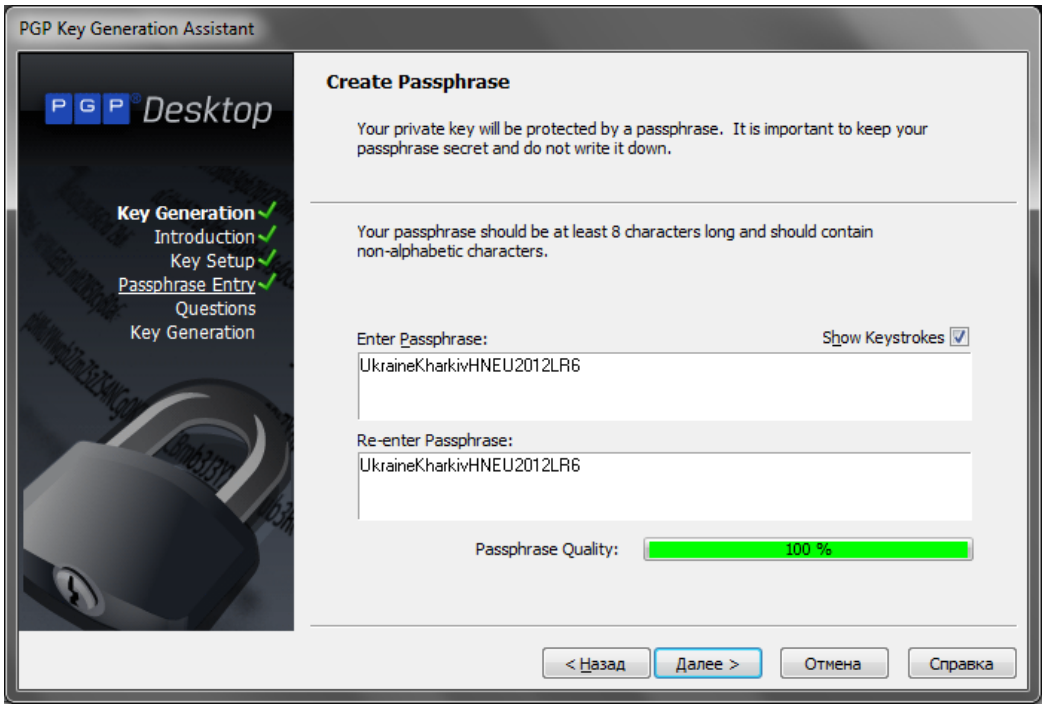


Рис. 6.6. **Створення контрольної фрази**

Після натискання кнопки "Далее" система згенерує ключ (рис. 6.7).

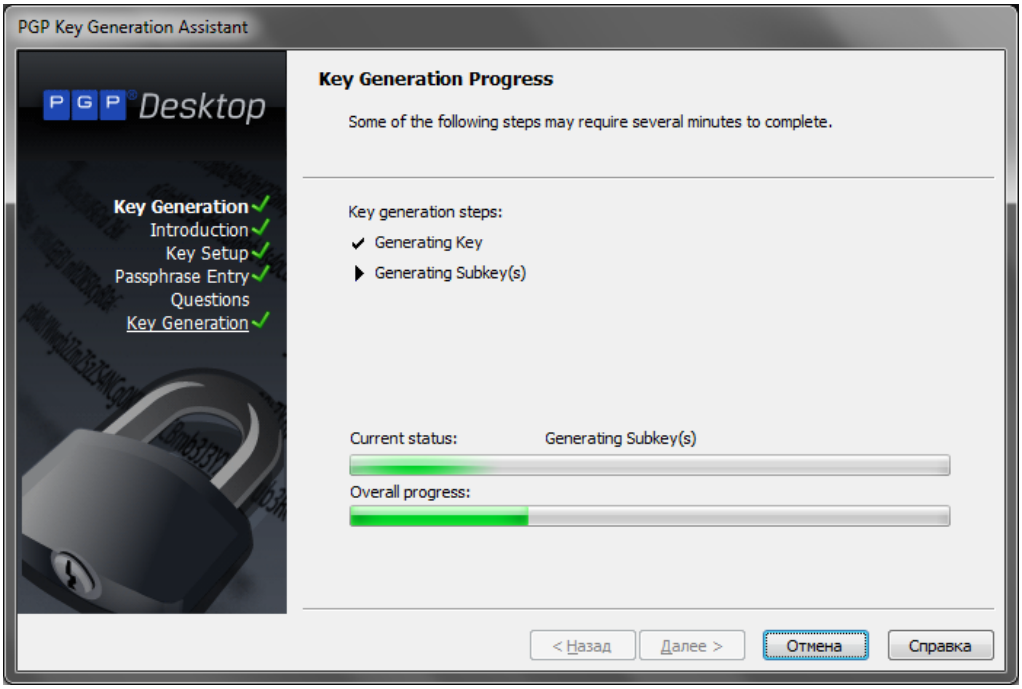


Рис. 6.7. **Генерування ключів**

Далі з'явиться вікно завершення формування ключів. Після натискання кнопки "Далее" відбудеться вихід із процесу створення ключа (рис. 6.8).

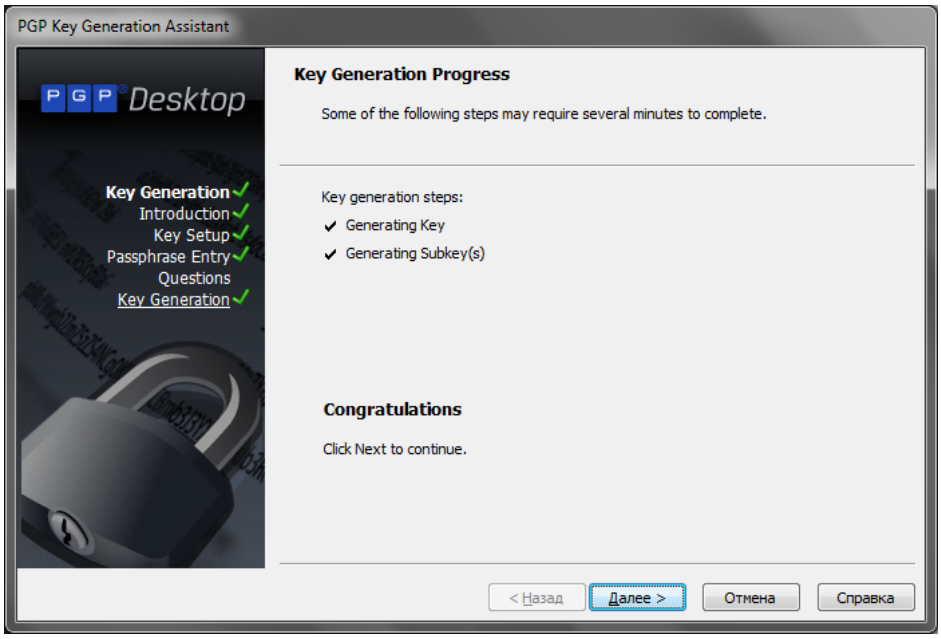


Рис. 6.8. **Формування ключа завершено**

Після створення ключа в полі праворуч відображають усі ключі, установлені у програмі (рис. 6.9). Необхідно сформувати зв'язку ключів і надіслати отримувачу свій публічний ключ. Після отримання публічного ключа збережіть його на *Рабочем столе* та проведіть імпортування в систему PGP.

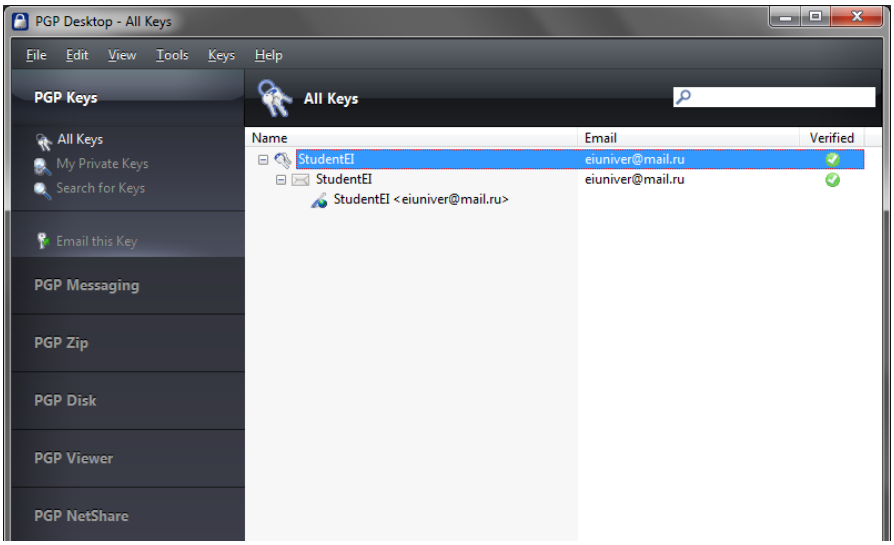


Рис. 6.9. **Список усіх установлених ключів**

Під час вибору пункту Key Properties у контекстному меню ключа відкривається вікно властивостей ключа (рис. 6.10). Для верифікації ключа виберіть Sign і встановіть ступінь довіри ключа відправника.

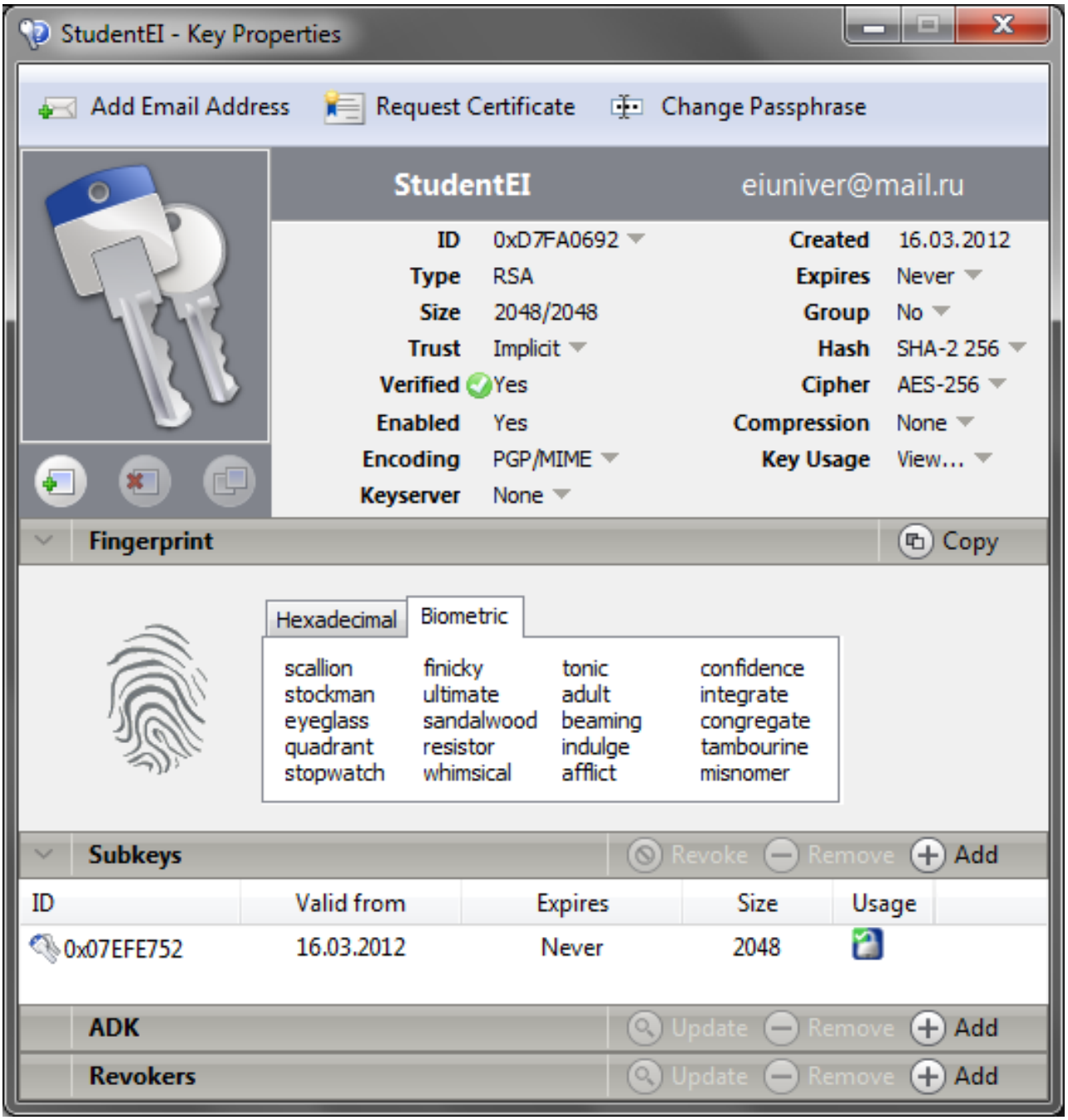


Рис. 6.10. **Властивості ключа**

Щоб отримувач міг розшифрувати зашифроване повідомлення, необхідно відіслати йому публічний ключ. Зробити це можна за допомогою пункту контекстного меню Export і вибору місця, куди зберегти файл ключа, або командою Send To -> Mail Recipient (рис. 6.11).

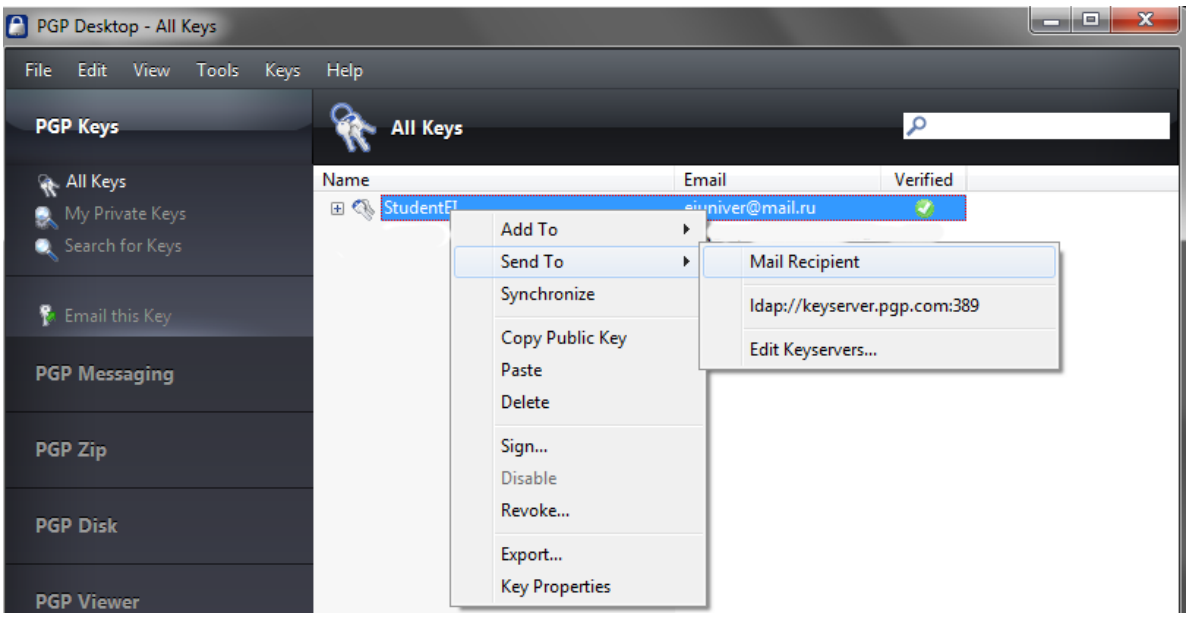


Рис. 6.11. **Відсилання ключа електронною поштою**

У цьому разі за замовчуванням відкриється вікно поштового клієнта для надсилання нового повідомлення, до якого прикріплено файл ключа (рис. 6.12).

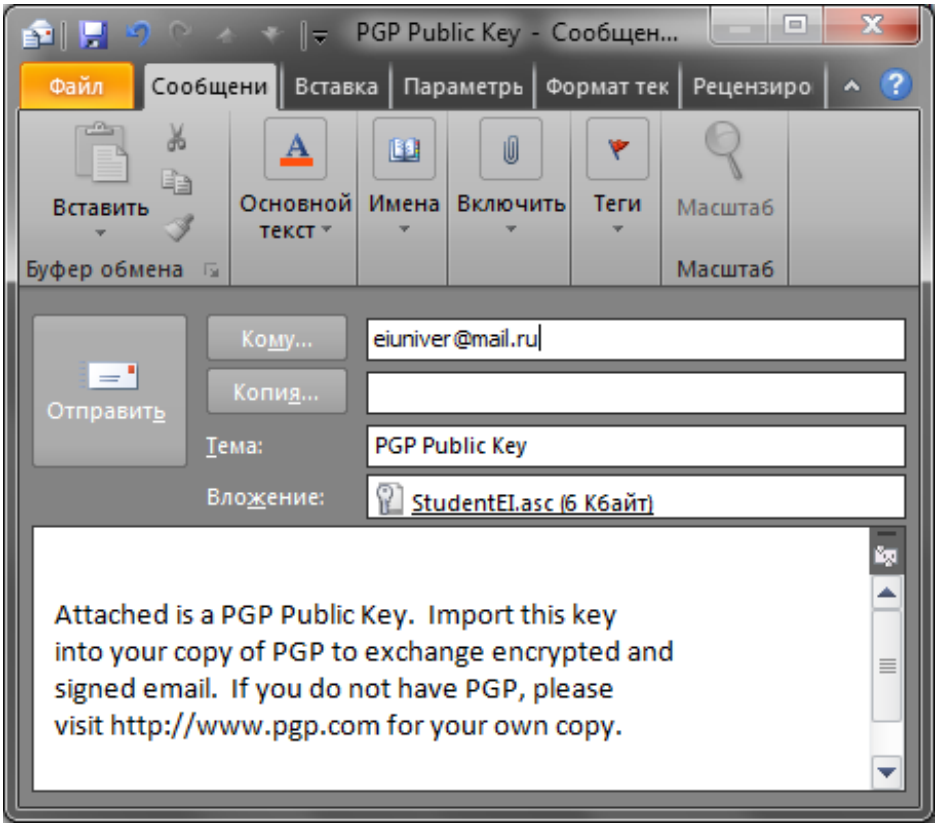


Рис. 6.12. **Експорт ключа електронною поштою**

Після завершення експорту ключа, отримуємо повідомлення із ключем (рис. 6.13).

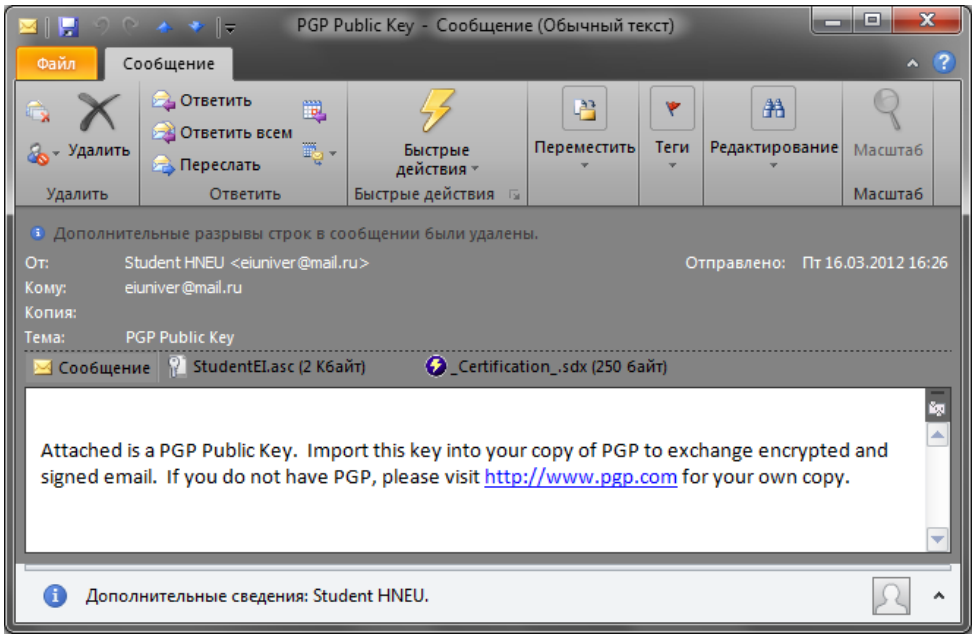


Рис. 6.13. **Отримане повідомлення із ключем**

Для створення зашифрованого повідомлення необхідно вибрати в контекстному меню програми пункт Clipboard -> Редагувати (рис. 6.14).

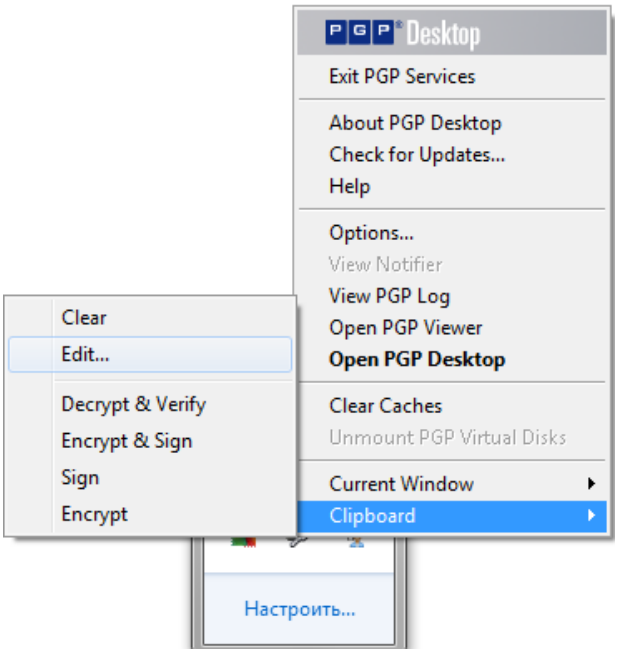


Рис. 6.14. **Створення захищеного повідомлення з допомогою PGP Desktop**

У вікно необхідно ввести текст для зашифрування та натиснути кнопку Copy to Clipboard (рис. 6.15). Після цього повідомлення з'явиться в буфері програми.

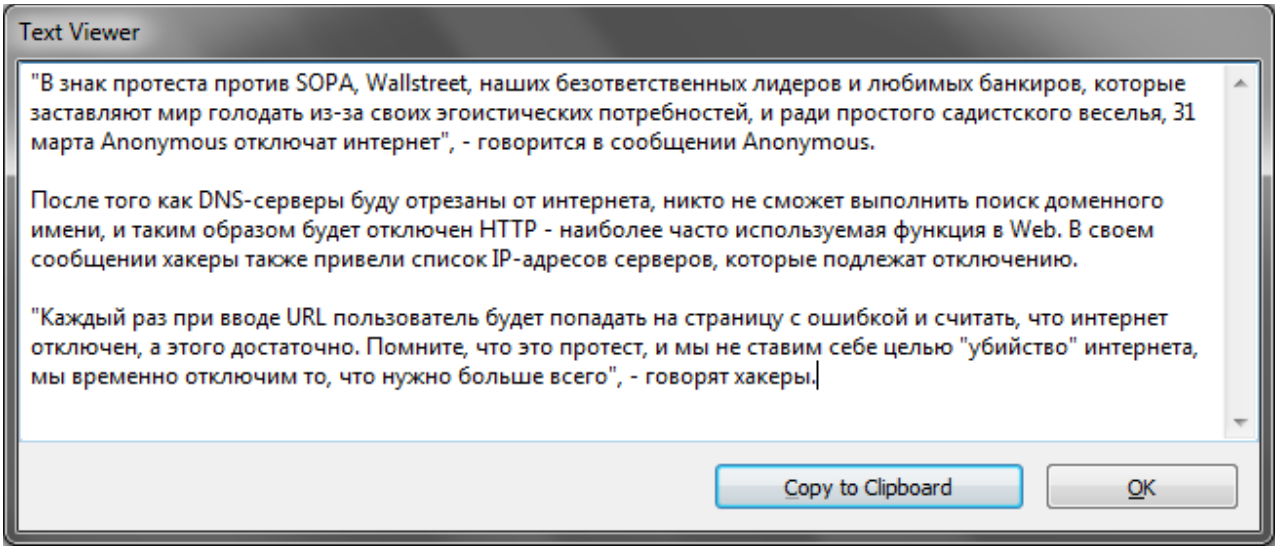


Рис. 6.15. **Вихідне повідомлення**

Щоб зашифрувати та підписати повідомлення, необхідно вибрати пункт Clipboard -> Encrypt&Sign (рис. 6.16). Для забезпечення конфіденційності (шифрування з використанням відкритого ключа отримувача використовують Clipboard -> Encrypt; для забезпечення автентифікації (шифрування з використанням особистого ключа відправника) використовують Clipboard -> Sign.

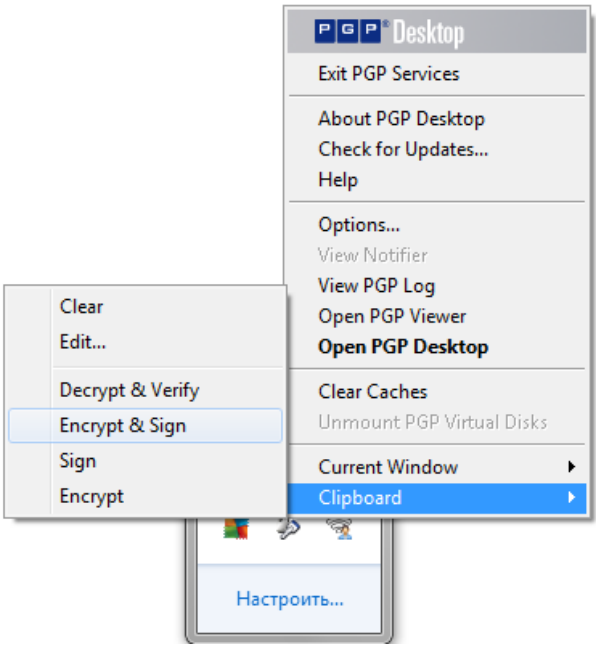


Рис. 6.16. **Вибір режиму захисту повідомлення**

Якщо цю операцію виконують уперше в цьому сеансі, то система зажадає вибрати ключ (рис. 6.17). Тому слід увести фразу-пароль, яку вказували для створення ключа.

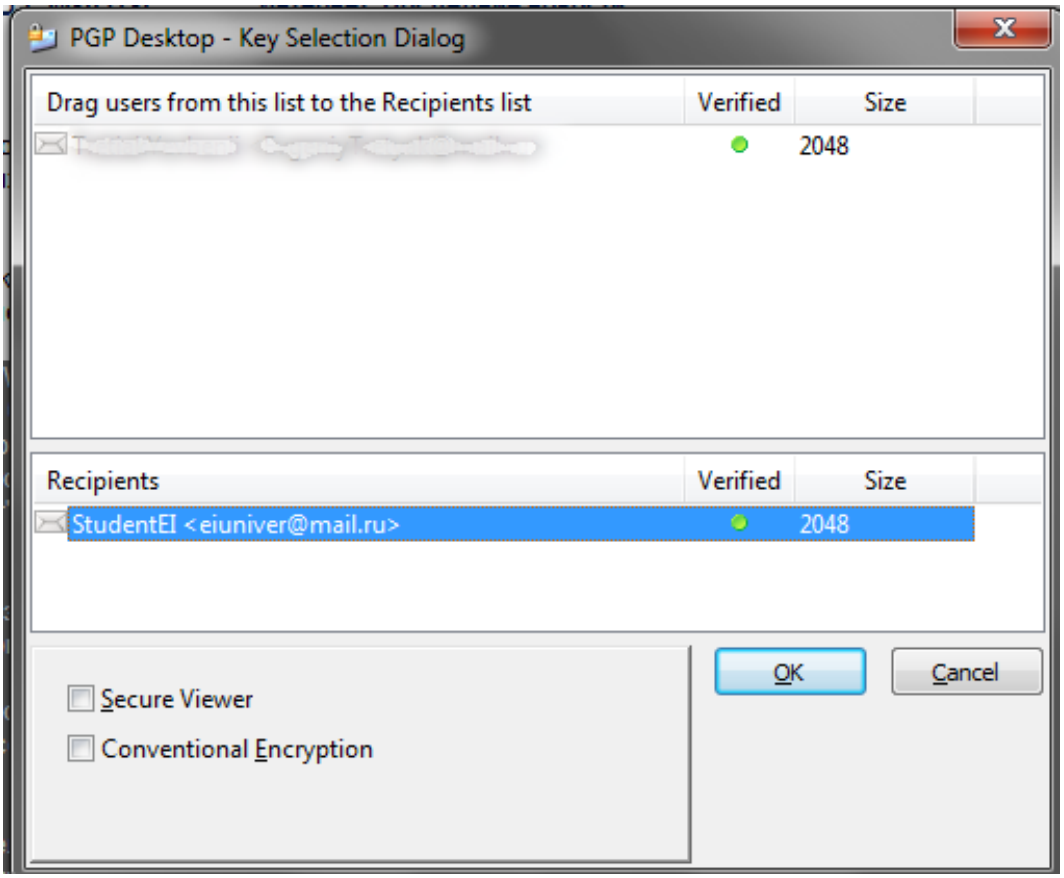


Рис. 6.17. **Вибір ключа для шифрування**

Після цього треба вибрати в контекстному меню програми пункт Clipboard -> Edit. Там буде розміщено зашифроване та підписане повідомлення.

Отримане повідомлення:

BEGIN PGP MESSAGE-----
Version: PGP Desktop 10.0.0 - not licensed for commercial use: www.pgp.com
Charset: utf-8
qANQR1DBwEwDkR8VpQfv51IBB/4rn3dfFsmiNkFcK21kaPgNMWciGi6h4rDONr1u
XDI3ZBuNnTpfdoQ827ti/52hByW7pkhgqw31AOQnZTeqXKOUPKloJHqxDMdnE4BK
oBmVN2F/7uTOU8wzBSSnM3U0mdzQlfrhC9R8ZlaSJK9NOJasecwcNSQ6Hy9gjPiB
3UaOY9hSBpGl4kcyM9FoU36sT7B67ZM/lIall6un1fdg+wGZXtj5kMad595DKovS
/A1MLc0109mUh0z+0YxJdOEWtbjeKh0roy0zKBJg6usALdP3z8Ivo3n8cJ1rOH34
v9CQEeYqC0H8bcNs/iPg3oUAhUkaty2i+cqWhMSMv4MuLaKi0sZQAf5wXb81SOSh
k0JnLCR2kvfAYdYA/iRl22azHZUPVjldDt/rdPZMTrhlKjyDcN0I0O7l6svYV0C/

Uj+seVrEC6frWgesNIST/qzaeS8HMkMG5fHNTzuWusO9o82tf1hZsnupeKsqQOkQ
R7Ua/q4f3D/QYey4I4kl415uAFg9WJdjT5L6BCULlEezVTw0E8kAr4xjOqfD4jqB
siA29TjXm8QBEGVAodgOxnImzfAeOHtsoOjh35ZL+uHEDSx4EWFkdghSdXFnhEqq
wEjF5W+AdLUX/vsWIw8BTWudR/u+xjUMXaozcX9IaUWu7m3xzC/EpnFYH68Wah8p
BlA5WpelbLk3XIJyhQALUixVYRWHiab2ry5jzESPr8xN+ABDjh/1OJvF0/4bvDGJ
f2mDZYBjdB/1ne2OhAUYKbmLwvktBR04MWso/r7qYhfTkCt6jhaXn8EQzQt3/IRv
...
UlOzTukrmjUSeWQ4DrS7tLVn0gvktys8aRtG4cY0niKNAIQQaus8HDQxXsUmj5h7
ahYeMcvcqu7crgZCCf84qt7Pb78ckFE==gLzv
-----END PGP MESSAGE-----

Для відновлення повідомлення необхідно вибрати пункт Clipboard -> Decrypt&Verify (рис. 6.18).

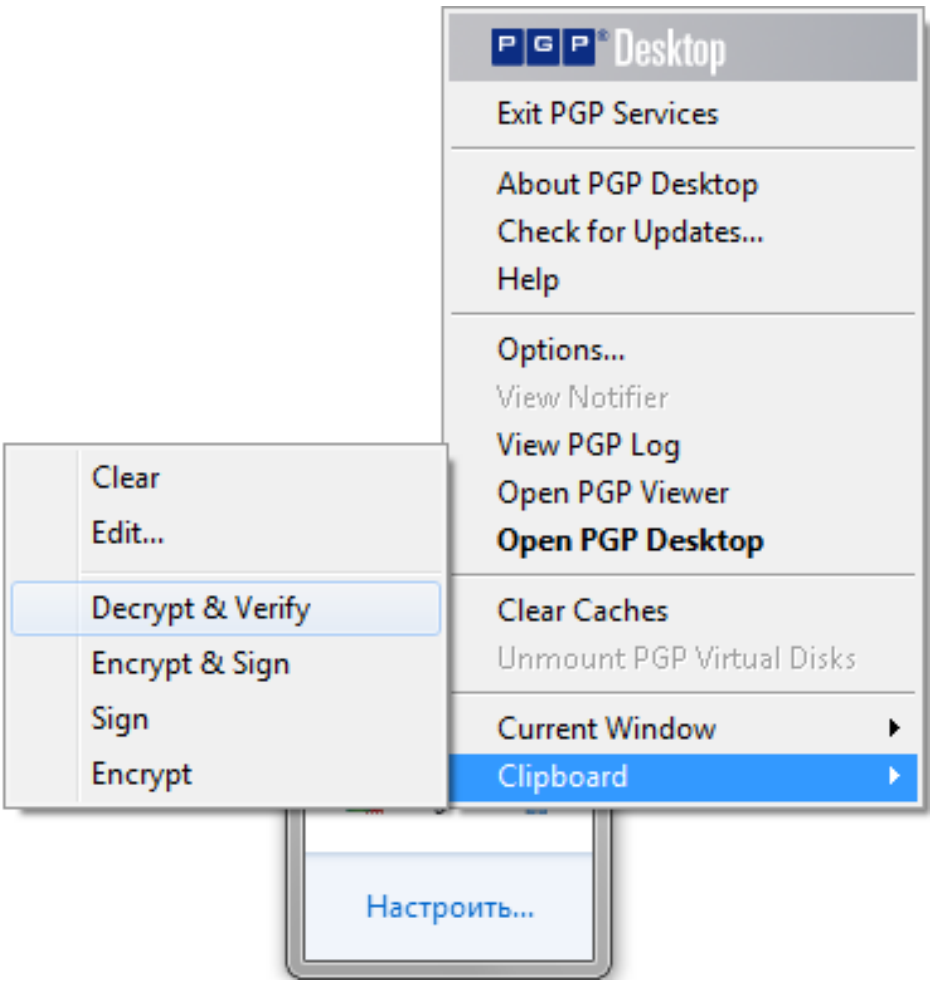


Рис. 6.18. **Відновлення повідомлення**

Потім зайти в пункт Clipboard -> Edit, де буде розміщено розшифроване повідомлення (рис. 6.19).

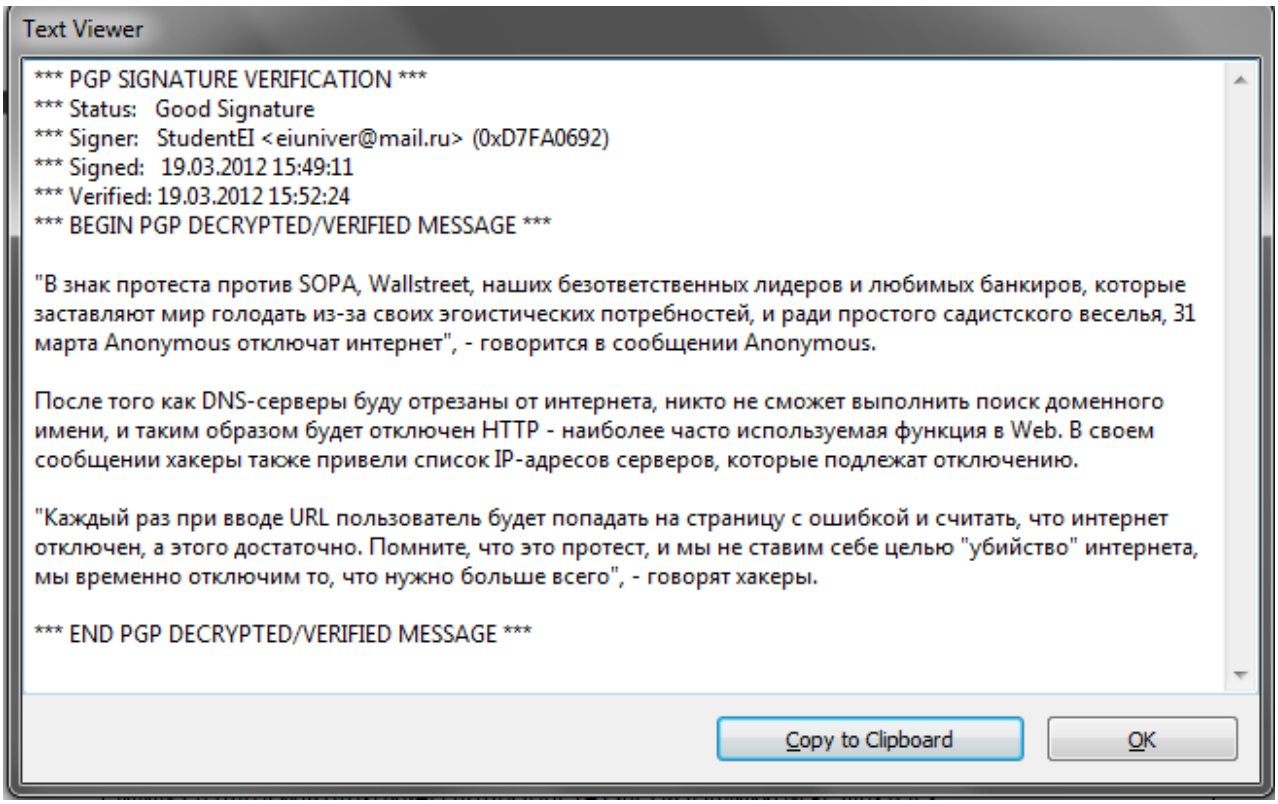


Рис. 6.19. **Відновлене повідомлення**

Щоб підписати це повідомлення, необхідно вибрати пункт Clipboard -> Sign. Після цього вибрати в контекстному меню програми пункт Clipboard -> Edit, буде підписане повідомлення.

Створене підписане повідомлення:

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

"В знак протеста против SOPA, Wallstreet, наших
безответственных лидеров и любимых
банкиров, которые заставляют мир
голодать из-за своих эгоистических
потребностей, и ради простого
садистского веселья, 31 марта Anonymous
отключат интернет", - говорится в
сообщении Anonymous.

После того как DNS-серверы буду отрезаны
от интернета, никто не сможет выполнить
поиск доменного имени, и таким образом
будет отключен HTTP - наиболее часто
используемая функция в Web. В своем
сообщении хакеры также привели список
IP-адресов серверов, которые подлежат
отключению.

"Каждый раз при вводе URL пользователь
будет попадать на страницу с ошибкой и
считать, что интернет отключен, а этого
достаточно. Помните, что это протест, и мы
не ставим себе целью "убийство"
интернета, мы временно отключим то, что
нужно больше всего", - говорят хакеры.
-----BEGIN PGP SIGNATURE-----
Version: PGP Desktop 10.0.0 - not licensed for commercial use: www.pgp.com
Charset: utf-8
wsBVAwUBT2c6VJ3Fdd3X+gaSAQgAdwf/QYGD836mOZVSfEuxYbrhF7DtNDn04Mez
5uwB/Z6CVTVl+0I3mwGhPJr3qXd4pxfD677R29Y7ogBczz8JYtPeesw/9MTONOrI
ril9aY5A3Znb7kPv2cXd5DDCNnQF8a627VFo4N/4W3e4xA/tw1Tbg0O4/d2kA5KL
...
=eWSS
-----END PGP SIGNATURE-----

Для перевірки підпису повідомлення необхідно вибрати пункт Clipboard -> Decrypt&Verify. Потім зайти в пункт Clipboard -> Edit, де буде розміщено повідомлення з результатом перевірки підпису (рис. 6.20).

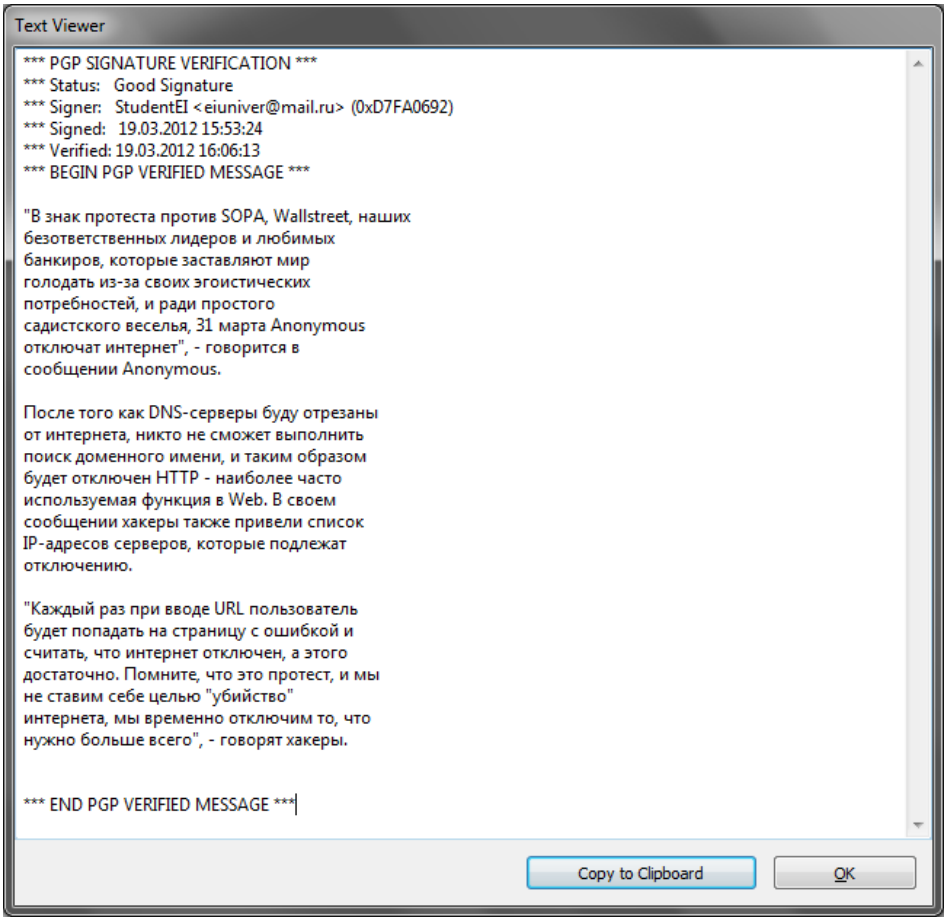


Рис. 6.20. **Перевірка підписаного повідомлення**

Щоб зашифрувати повідомлення без підпису необхідно вибрати пункт Clipboard -> Encrypt. Після цього, вибрати в контекстному меню програми пункт Clipboard -> Edit.... Там буде зашифроване повідомлення:

Зашифроване повідомлення без підпису:

```
-----BEGIN PGP MESSAGE-----
Version: PGP Desktop 10.0.0 - not licensed for commercial use: www.pgp.com
Charset: utf-8
qANQR1DBwEwDkR8VpQfv51IBB/4iJSFa/HSeibGdtu+RVy7RNw5rsrn6dvUDMUqH
u0/o4UwdPo2mplK4eDFNFfwBmhjkiuLzuS5C5yI8hUbEfDsunGYkF1MqjA0x2UGz
A/p4eWp0G5pRlS5dZVeFoa+V+4rBmiCIB9HFYxKW+3byuqCm0x4wXgMbdpgjTV0V
BIThrmOCMM2N1E6kQfHAFQ==
=t0N1
-----END PGP MESSAGE-----
```

Для відновлення повідомлення необхідно вибрати пункт Clipboard -> Decrypt&Verify. Потім зайти в пункт Clipboard -> Edit, де буде розміщено розшифроване повідомлення (рис. 6.21).

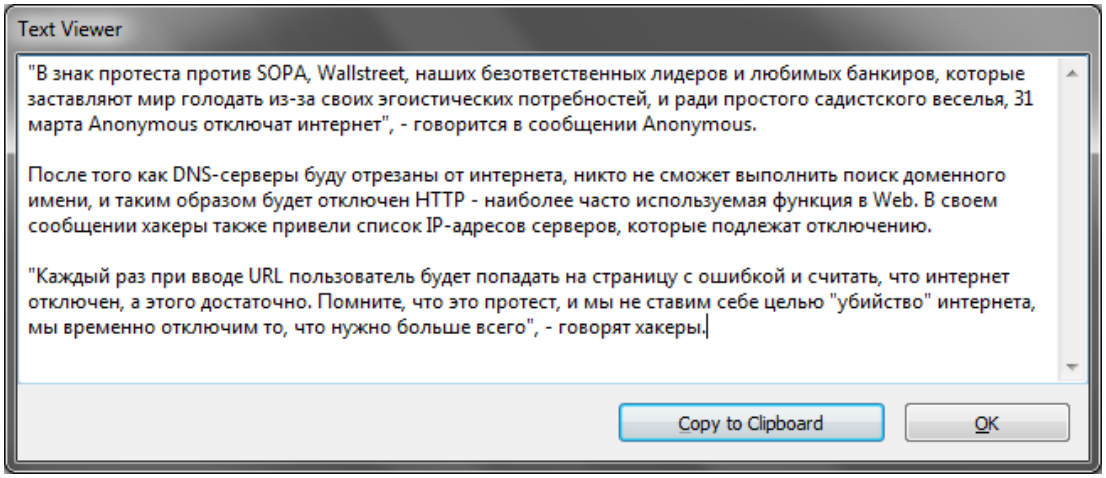


Рис. 6.21. **Відновлене повідомлення**

6.6. Зміст звіту

Опишіть:

- порядок використання публічних і приватних ключів;
- шифрування та розшифрування електронних повідомлень;
- цифрового підпису та перевірки вірогідності джерела повідомлення;
- шифрування та цифрового підпису;
- розшифрування та перевірки підпису.

6.7. Контрольні запитання

1. Які функції виконує система PGP?

2. Що таке "автентичність", як реалізовано цей механізм у системі PGP?

3. Що таке "конфіденційність", як реалізовано цей механізм у системі?

4. Що таке "цифровий підпис", як реалізовано цей механізм у системі?

5. Для чого призначено схему Діффі – Геллмана в системі PGP?

6. Що таке "модель довіри"?

7. Як формують та розподіляють ключі в системі PGP?

8. Яким чином здійснюють протокол обміну інформацією між користувачами для забезпечення конфіденційності на основі схеми PGP (відповідно до структурної схеми)?

9. Яким чином здійснюють протокол обміну між користувачами для забезпечення автентичності на основі схеми PGP (відповідно до структурної схеми)?

10. Яким чином здійснюють протокол обміну між користувачами для забезпечення конфіденційності й автентичності на основі схеми PGP (відповідно до структурної схеми)?

Лабораторна робота 7
Статистичні дослідження генераторів випадкових та псевдовипадкових послідовностей за методикою NIST STS

7.1. Мета

Метою лабораторної роботи є вивчення методики досліджень статистичних властивостей генераторів випадкових і псевдовипадкових послідовностей.

7.2. Рекомендації до підготовки до виконання

Під час підготовки та проведення лабораторної роботи студенти мають:

- закріпити теоретичні знання про стандартні методики тестування випадкових і псевдовипадкових послідовностей;
- навчитися проводити статичні дослідження за допомогою спеціалізованого програмного забезпечення й автоматизованих робочих місць із тестування;
- набути практичних навичок в аналізі результатів статистичного експерименту;
- закріпити теоретичні знання про застосування критеріїв погодженості та перевірки гіпотез щодо закону розподілу випадкових величин.

7.3. Загальні теоретичні положення

Контроль за ефективністю функціонування генераторів випадкових та псевдовипадкових послідовностей. Генератор випадкових послідовностей (ГВП) як програмно-апаратний засіб має певний життєвій цикл, який містить етапи проєктування, упровадження, експлуатації утилізації. Відповідно, увесь процес оцінювання ефективності та контролю за нею його функціонування ГВП необхідно прив'язати до етапів життєвого циклу.

На етапі проєктування та впровадження ГВП проводять глибокі та детальні дослідження ефективності його функціонування. Для цього етапу характерна відсутність жорстких часових меж для здійснення перевірок властивостей та характеристик ГВП. Важливо дістати якомога повну інформацію про функціонування ГВП, необхідну для ухвалення рішення про можливість використання генератора в системах криптографічного захисту інформації. На етапі впровадження системи криптографічного захисту детальні дослідження властивостей ГВП є складовою частиною досліджень, які проводять для ухвалення рішення щодо видавання дозволу на використання криптографічної системи захисту інформації.

Це завдання вирішують за допомогою програмно-технічного комплексу статистичного тестування та перевірки працездатності ГВП. Основною метою комплексу є проведення детальних статистичних випробувань та здійснення комплексного контролю за функціонуванням ГВП.

Метою *комплексного контролю* є оцінювання ефективності функціонування, дослідження властивостей випадкових послідовностей та вимірювання характеристик і показників ГВП. Комплексний контроль мають здійснювати на основі методик статичного тестування, а також промислового випробування на спеціалізованому стенді.

На етапі експлуатації системи захисту інформації, до складу якої входять ГВП, необхідно здійснювати оперативний та поточний контроль за справністю ГВП. Із цією метою у склад ГВП або системи захисту інформації вбудовують засоби оперативного та поточного контролю за ефективністю його функціонування.

Мета *оперативного контролю* – визначення стану АМ (ГВП) формування випадкових чисел та якості випадкових послідовностей. Оперативний контроль здійснюють після кожного процесу формування випадкових послідовностей. Тому необхідно будувати засоби оперативного контролю з урахуванням жорстких обмежень на час здійснення контролю й інші ресурси (пам'ять, процесорне завантаження та ін.). У зв'язку із цим підсистема оперативного контролю має здійснювати також засоби швидкого статистичного контролю. Завдання оперативного контролю – не допустити використання криптографічно слабких, заборонених параметрів для виконання криптографічних перетворень. Для реалізації оперативного контролю програмне забезпечення ГВП має містити засоби статичного контролю.

Метою *поточного контролю* є оцінювання працездатності ГВП, а також своєчасне виявлення будь-яких порушень його функціонування. Поточний контроль здійснюють, відповідно до планових перевірок у межах технічного обслуговування системи криптографічного захисту. Поточний контроль здійснюють із використанням більш широкого, порівняно з оперативним контролем, набору засобів. У ході поточного контролю виконують статистичне тестування випадкових послідовностей, оцінювання властивостей послідовностей, вимірювання технічних контрольних параметрів. У зв'язку з обмеженням часу на здійснення поточного контролю можливе використання скорочених наборів тестів.

Таким чином, для оцінювання ефективності та контролю за працездатністю ГВП необхідно використовувати комплексну, поточну та оперативну форми контролю.

Комплексний контроль мають здійснювати тільки із застосуванням автоматизованого робочого місця або спеціалізованого стенду. На рис. 7.1 надано структурну схема програмно-апаратного комплексу тестування.

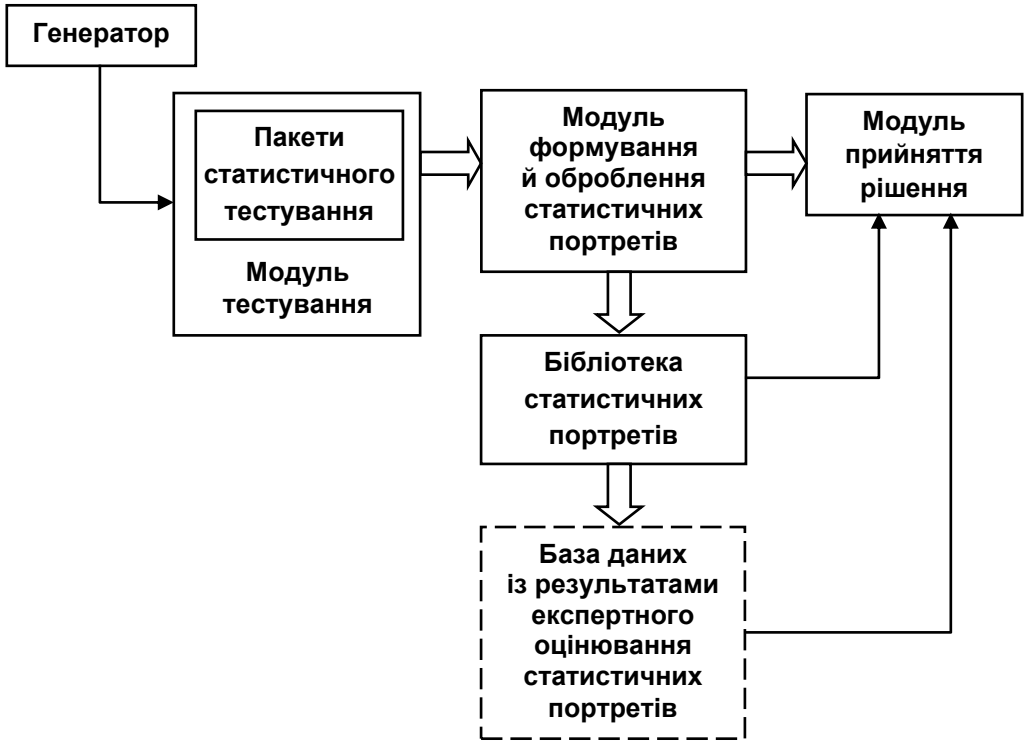


Рис. 7.1. **Структурна схема програмно-апаратного комплексу тестування**

До складу комплексу входять:

- модулі формування випадкових або псевдовипадкових послідовностей (програмний, програмно-апаратний та апаратний генератор випадкових послідовностей), спеціалізоване програмне забезпечення управління генератором, засоби контролю за працездатностю генератора та інше обладнання;
- модуль статистичного тестування – прикладне спеціалізоване програмне забезпечення, що реалізує методику статистичного тестування;
- модуль аналізу статистичних портретів візуалізації результатів випробувань;
- електронна бібліотека результатів тестування;
- засоби підтримання прийняття рішень.

Комплекс доцільно реалізувати у вигляді автоматизованого робочого місця на базі ПЕОМ. У такому разі можливий як комплексний, так і поточний та оперативний контроль ГВП. Модуль статистичного тестування реалізує методику та набір статистичних тестів, його реалізують у вигляді прикладного програмного забезпечення. Результати тестування потребують певного оброблення та візуалізації, тому необхідно реалізувати функції візуалізації результатів. Такі функції можуть бути як убудованими до модуля тестування, так і реалізовані за допомогою відомих пакетів оброблення даних, наприклад MS Excel, Statistica та ін. (див. додаток А).

Електронна бібліотека необхідна для накопичення результатів тестування, із метою подальшого використання результатів для порівняльного аналізу нових результатів, а також для надання інформації ухвалення більш обґрунтованих рішень щодо випробувань. До складу бібліотеки входять і контрольні (тестові) результати.

Основні вимоги до ПЕОМ і програмного забезпечення:

- операційна система – від Windows XP;
- тактова частота процесора – бажано, не нижча за 750 Мгц. Це обумовлено тим, що здійснення комплексного контролю потребує часу (так, випробування за NIST STS проводять протягом 1 год);
- до складу прикладного забезпечення входять програмне забезпечення тестування, програмне забезпечення управління ГВП, офісні додатки.

7.4. Критерії прийняття рішень щодо випадковості послідовності

Найбільш поширеним на практиці підходом до визначення псевдовипадковості є евристичний підхід. За цього підходу псевдовипадковий генератор розглядають як програму (алгоритм), яка породжує бітову послідовність $s = s_0, s_1, \ldots, s_{n-1}$ скінченої довжини n, що проходить деякі особливі статистичні тести. Таким чином, властивості випадкової або псевдовипадкової послідовності може бути охарактеризовано й описано в імовірнісному сенсі.

Є безліч тестів, які визначають чи є послідовність випадковою. Але жоден закінчений кінцевий набір тестів не є достатнім. Крім того, результати статистичного тесту мають інтерпретувати з деякою обережністю та застереженням, для того щоб уникнути хибних висновків за певним генератором.

Статистичний тест формулюють для перевірки певної нульової гіпотези H_0 про те, що послідовність є випадковою. Із цією нульовою гіпотезою пов'язано альтернативну гіпотезу H_a, що послідовність не є випадковою. Для кожного тесту можна зробити висновок про прийняття або відхилення нульової гіпотези виходячи, зі сформованої генератором послідовності.

Для кожного тесту має бути вибрано адекватну статистику випадковості, на основі якої може бути прийнято або відхилено нульову гіпотезу. Згідно із припущенням про випадковість, така статистика має розподіл випадкових значень. Теоретично для нульової гіпотези розподілу цієї статистики визначено математичними методами. Із цього еталонного розподілу визначено критичне значення. Під час проведення тесту розраховують значення тестової статистики. Це значення порівнюють із критичним значенням. Якщо значення тестової статистики перевищує критичне значення, нульову гіпотезу для випадковості відхиляють. Інакше нульову гіпотезу приймають.

Перевірка статистичних гіпотез працює завдяки тому, що еталонний розподіл та критичне значення залежать і їх генерують, відповідно до попереднього припущення про випадковість. Якщо припущення про випадковість – істина, то результат тестової статистики для неї буде мати дуже низьку ймовірність перевищення критичного значення (наприклад 0,01). Якщо розрахункове значення тестової статистики перевищує

критичне значення (тобто виникає подія з низькою ймовірністю), то в аспекті перевірки статистичної гіпотези подія з низькою ймовірністю не може бути як така. Тому, якщо розрахункове значення тестової статистики перевищує критичне, роблять висновок, що перше припущення щодо випадковості є недоцільним або помилковим. У цьому разі роблять висновок про відхилення H_0 (випадковість), і прийняття H_a (не випадковість). Перевірка статистичної гіпотези є процедурою генерації висновків, під час виконання якої можливе прийняття H_0 (дані випадкові) або відхилення H_0 (дані невипадкові). Табл. 7.1 пов'язує істинний (невідомий) стан даних із висновком, визначеним процедурою перевірки.

Таблиця 7.1

Зв'язок стану даних із висновком процедури перевірки

Ситуації	Висновки	
	Прийняти H_0	Прийняти H_a
Дані випадкові (H_0 – істина)	Немає помилки	Помилка I роду
Дані не випадкові (H_a – істина)	Помилка II роду	Немає помилки

Якщо дані дійсно є випадковими, то висновок про відхилення нульової гіпотези будуть приймати дуже рідко. Цей висновок має назву помилки I роду. Якщо дані не є випадковими, то висновок про прийняття нульової гіпотези (тобто дані випадкові) має назву помилка II роду. Висновок про прийняття H_0, коли дані дійсно є випадковими, і відхилення H_0, коли дані не є випадковими, є правильним.

Імовірність помилки I роду називають рівнем значущості тесту. Цю ймовірність може бути встановлено до іспитів і позначають її як α. Для тесту сутність α полягає в тому, що тест покаже на невипадковість послідовності, тоді як, насправді, вона є випадковою. Тобто послідовність має невипадкові властивості, навіть коли її сформував "гарний" генератор.

Імовірність помилки II роду позначають як β. Для тесту β – імовірність того, що тест покаже на випадковість послідовності, коли, насправді, вона не є випадковою. Тобто "поганий" генератор сформував послідовність, яка, як здається, має випадкові властивості. На відміну від α, помилка II роду β не є фіксованим значенням. Вона може набувати множину

різних значень, тому що є безліч ситуацій, коли потік даних може бути не випадковим, і кожна з них видає різні β. Обчислення помилки II роду більш складні із-за великої кількості можливих типів невипадковості.

Для прийняття рішень про проходження послідовністю випадкових чисел статистичного тесту використовують три основні підходи.

Нехай задано двійкову послідовність $S = \{s_1, s_2, \ldots, s_n\}$, $s_i \in \{0, 1\}$ довжиною n бітів. Необхідно ухвалити рішення, проходить ця послідовність статистичний тест на випадковість чи ні. Можливі такі підходи до розв'язання цієї задачі.

1. Критерій прийняття рішення на основі встановлення граничного рівня. Цей підхід ґрунтується на обчисленні статистики тесту $c(S)$ із її порівнянням із деяким граничним рівнем $c_{пор.}(S)$. Критерій прийняття рішення формують таким чином: *уважають, що двійкова послідовність S не проходить статистичного тесту кожного разу, коли статистика тесту c(S) набуває значень, менших за граничний рівень* $c_{пор.}(S)$.

Наприклад, у процесі складності послідовності з використанням тесту на основі алгоритму Лемпеля – Зіва, для заданої двійкової послідовності S розраховують її складність $c(S)$. Для того щоб визначити, пройшла ця послідовність тест чи ні, необхідно порівняти визначене значення $c(S)$ із граничним значенням $n/\log_2 n$. Однак такий підхід не є достатньо надійним. Як показали практичні дослідження, використання такого критерію часто призводить до помилкових рішень.

2. Критерій прийняття рішень на основі встановлення фіксованого довірчого інтервалу. За такого підходу критерій прийняття рішення формують таким чином*: уважають, що двійкова послідовність S не проходить статистичного тесту, якщо значення статистики тесту c(S) перебуває за межею довірчого інтервалу значень статистики обчисленого для встановленого рівня значущості* α. Наприклад, нехай до двійкової послідовності S довжиною $n = 800$ бітів застосовують частотний тест. Значення статистики тесту $c(S)$ – це кількість одиниць у послідовності S, причому очікують, що в послідовності буде приблизно 400 одиниць і 400 нулів. Якщо зафіксувати рівень значущості приблизно 5 % ($\alpha = 0{,}05$), то послідовність S не пройде частотного тесту, якщо кількість одиниць буде перебувати за межами довірчого інтервалу:

$$400 \pm \frac{1{,}96}{2} \times \sqrt{800} = [373{,}427].$$

Цей критерій, порівняно з першим, є більш надійним. Необхідно тільки враховувати, що різним рівням значущості будуть відповідати різні довірчі інтервали.

3. Третій підхід побудови критерію прийняття рішення спирається на розрахунок для статистики тесту $c(S)$ відповідного значення ймовірності P-value. Тут статистику тесту розглядають як реалізацію випадкової величини, яка підкоряється відомому закону розподілу. Статистику тесту будують таким чином, щоб її менші значення вказували на будь-який дефект випадковості послідовності. Значення P-value є ймовірністю того, що статистика тесту набуде значення, більшого за те, що спостерігають під час випробування випадкової послідовності (у передабачені її випадковості). Таким чином, малі значення P-value (P-value < 0,05 або P-value < 0,01) інтерпретують як доказ того, що послідовність не є випадковою. Вирішальне правило формують так: *для фіксованого рівня значущості α двійкова послідовність S не проходить статистичного тесту, якщо значення ймовірності P-value < α. Значення α рекомендовано вибирати з інтервалу [0,001; 0,01]*.

Значення α, дорівняне 0,001, свідчить, що з 1 000 випадкових послідовностей тест може не пройти лише одна. Із P-value ≥ 0,001 послідовність розглядається як випадкова із довірою 99,9 %. З P-value < 0.001 послідовність розглядають як не випадкову з довірою 99,9 %.

Значення α, дорівняне 0,01, свідчить про те, що зі 100 випадкових послідовностей не пройшла б тесту лише одна. Із P-value ≥ 0,01 послідовність розглядають як випадкову з довірою 99 %. Із P-value < 0,01 послідовність розглядають як не випадкову з довірою 99 %.

Однією з основних цілей тестів, які будують за третім підходом є мінімізація ймовірності помилки II роду, тобто мінімізація ймовірності прийняття послідовності, сформованої "поганим" генератором, за послідовність, сформовану "гарним" генератором. Імовірності α, β пов'язано одна з одною та з довжиною n послідовності, що перевіряють: якщо два із цих значень визначено, третє визначається автоматично. На практиці зазвичай вибирають розмір n і значення для α (імовірність помилки I роду). Тоді критичну точку вибирають таким чином, щоб знайти найменше значення β (імовірність помилки II роду).

Таким чином, основним підходом, який можна використовувати в дослідженні властивостей, є евристичний. Однією з важливих завдань застосування евристичного підходу на практиці є обґрунтування набору

статистичних тестів. Склад тестів залежить від призначення генератора та способів використання послідовностей. Із філософського погляду немає можливості евристичними методами довести випадковість послідовності. Тому конкретна послідовність має проходити континуум тестів. Але й тоді не можна стверджувати, що така послідовність випадкова, тому що можливе створення нового тесту, який вона не пройде. Через це неможливо також і побудувати універсальні тести, наприклад універсальний тест Маурера. Як показали дослідження, можливі варіанти, коли такі тести проходять і невипадкові послідовності.

На сьогодні є декілька визначених методик статистичного тестування іноземних і вітчизняних фахівців.

У табл. 7.2 наведено інформацію про відомі системи статистичних тестів.

Таблиця 7.2

Системи статистичних тестів

Розробники / назви тестів	Посилання на опис
Д. Кнут (Стенфордський університет, США)	Мистецтво програмування. Напівчислові алгоритми
Дж. Марсалья (Флоридський державний університет, США)	Система статистичного тестування DIEHARD
Х. Густавсон та ін. (Куїндсландський технологічний університет, Австралія)	Система CRYPT-S
І. Горбенко, О. Потій (Харківський військовий університет, Україна)	Методика статистичного тестування генераторів псевдовипадкових послідовностей
Методика тестування FIPS 140-2	NIST PUB FIPS 140-2
Методика NIST (США)	NIST Statistical test Suite
Методика тестування RIPE	www.nessie.org

7.5. Методика тестування NIST STS

Загальні положення. Набір тестів NIST STS було запропоновано в ході проведення конкурсу на новий національний стандарт США для блокового шифрування. Цей набір використовували для досліджень статистичних властивостей кандидатів на новий блоковий шифр. На сьогодні методика тестування, запропонована NIST, є найбільш поширеною у розробників криптографічних засобів захисту інформації.

Порядок тестування окремої двійкової послідовності S має такий вигляд:

1) висувають нульову гіпотезу H_0 – припущення про те, що ця двійкова послідовність S є випадковою;

2) за послідовністю S розраховують статистику тесту $c(S)$;

3) із використанням спеціальної функції та статистики тесту розраховують значення ймовірності $P = f\left(c(S)\right)$, $P \in [0,1]$;

4) значення ймовірності P порівнюють із рівнем значущості α, $\alpha \in [0{,}001;\ 0{,}01]$. Якщо $P \geq \alpha$, то гіпотезу H_0 приймають. Інакше приймають альтернативну гіпотезу.

Пакет містить 16 статистичних тестів. Але, залежно від вхідних параметрів, обчислюють 189 значень імовірності P, які можна розглядати як результат роботи окремих тестів. У табл. 7.3 наводять зібрані з усіх тестів із вказівкою на кількість обчислювальних значень імовірності P, фізичного змісту статистики тесту та дефекту, на виявлення якого спрямовано тест.

Таблиця 7.3

Статистика тестів та дефекти, що виявляють

№ п/п	Статистичні тести	Статистика тесту $c(S)$	Дефекти, що виявляють
1	2	3	4
1	Частотний (монобітний тест)	Нормалізована абсолютна сума значень елементів послідовності	Надто багато нулів або одиниць у послідовності
2	Частотний тест (у середині блока)	Міра погодженості кількості одиниць, спостережених із тим, що очікують теоретично	Локалізовані відхилення частоти появи одиниць у блоці від ідеального значення ½
3	Перевірка накопичених сум	Максимальне відхилення значень накопиченої суми елементів послідовності від початкової точки відліку (точка 0)	Велика кількість одиниць або нулів на початку або наприкінці двійкової послідовності
4	Перевірка серій	Загальна кількість серій на всій довжині послідовності	Надто швидка або надто повільна зміна знака в ході генерації послідовності

Продовження табл. 7.3

1	2	3	4
5	Перевірка максимальної довжини серії у блоці	Міра погодженості значень максимальної довжини, спостережених зі значенням, що очікують теоретично	Відхилення від теоретичного закону розподілу максимальних довжин серій одиниць
6	Перевірка рангу двійкової матриці	Міра погодженості значення рангів різного порядку, спостережених зі значенням, що очікують теоретично	Відхилення емпіричного закону розподілу значень рангів матриць від теоретичного, що вказує на залежність символів у послідовності
7	Спектральний аналіз на основі дискретного перетворення Фур'є	Нормалізована різниця кількості спостережуваних частотних компонент, що очікують (які перевищують 95 % рівень порогу	Виявлення періодичних складових (трендів) у двійковій послідовності
8	Перевірка шаблонів, що перекривають	Міра погодженості кількості шаблонів, що перекривають, у послідовності із теоретичним значенням	Велика кількість m-бітових серій з одиниць у послідовності
9	Універсальний тест Маурера	Сума логарифму відстані між l-бітовими шаблонами	Можливість стиснення послідовності
10	Ентропійний тест	Міра погодженості значення ентропії джерела з тим, що теоретично очікують для випадкового джерела	Нерівномірність розподілу m-бітових слів у послідовності (регулярність властивостей джерела)
11	Перевірка випадкових відхилень	Міра погодженості кількості візитів за випадкового блукання в заданий стан у середині циклу з тим, що очікують теоретично	Відхилення від теоретичного закону розподілу візитів у конкретний стан за випадкового блукання
12	Перевірка випадкових відхилень (варіант)	Загальна кількість візитів за випадкового блукання	Відхилення від теоретично очікуваної кількості візитів за випадкового блукання в заданий стан від теоретичного
13	Послідовний тест	Міра погодженості кількості всіх варіантів m-бітових шаблонів із тією, що очікують теоретично	Нерівномірність розподілу m-бітових слів у послідовності

Закінчення табл. 7.3

1	2	3	4
14	Перевірка стиснення, згідно з алгоритмом Лемпеля – Зіва	Кількість у послідовності різних слів	Великий ступінь стиснення послідовності, порівняно зі ступенем стиснення, що очікують для випадкової послідовності
15	Перевірка шаблонів, що не перекривають	Міра погодженості кількості неперіодичних шаблонів у послідовності з теоретичним значенням	Велика кількість заданих неперіодичних шаблонів у послідовності
16	Перевірка лінійної складності	Міра погодженості кількості подій, що полягають у появі фіксованої довжини еквівалентного ЛРР для заданого блока з теоретичним	Відхилення емпіричного розподілу довжин, еквівалентних ЛРР, для послідовностей фіксованої довжини від теоретичного закону розподілу для випадкової послідовності, що вказує на недостатню складність тестованої послідовності

Таким чином, у результаті тестування двійкової послідовності формують вектор значень імовірності $P = \{P_1, P_2, \ldots, P_{189}\}$. Аналіз складових P_i цього вектора дозволяє вказати на конкретні дефекти випадковості тестованої послідовності.

Методика має такий вигляд:

1. Для кожного ГВП необхідно оцінити та прийняти рішення про те, що він формує випадкові двійкові послідовності. Генератор має формувати двійкову послідовність $S = \{s_1, s_2, \ldots, s_n\}$, $s_i \in \{0,1\}$ довільної довжини n.

2. Для фіксованого значення n формують множину з m двійкових послідовностей:

$$\begin{aligned} S1 &= \{s_1, s_2, \ldots, s_n\}; \\ S2 &= \{s_1, s_2, \ldots, s_n\}; \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ Sm &= \{s_1, s_2, \ldots, s_n\}. \end{aligned} \qquad (7.1)$$

Таким чином, для тестування необхідно сформувати вибірку обсягом $N = m \times n$.

3. Кожну послідовність перевіряють із використанням пакета NIST STS. У результаті формують статистичний портрет генератора виду (рис. 7.2).

№ тесту j / № послідовності i	1	2	…	q
S_1	$P_{1,1}$	$P_{1,2}$		$P_{1,q}$
S_2	$P_{2,1}$	$P_{2,2}$		$P_{2,q}$
:	:			
S_m	$P_{m,1}$	$P_{m,2}$		$P_{m,q}$

$$\Rightarrow \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1q} \\ P_{21} & P_{22} & \cdots & P_{1q} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mq} \end{pmatrix}$$

Рис. 7.2. **Статистичний портрет генератора за методикою NIST STS**

Статистичним портретом генератора є матриця розмірністю $m \times q$, де: m – кількість двійкових послідовностей, що перевіряють, а q – кількість статистичних тестів, що використовують для тестування кожної послідовності. Елементи матриці $P_{ij} \in [0,1]$ для $i = \overline{1.m}$, $j = \overline{1,q}$ – це значення ймовірності, визначену в результаті тестування i-ї послідовності j-м тестом.

4. За визначеним статистичним портретом визначають частку послідовностей, що пройшли кожний статистичний тест. Для цього задають рівень значущості $\alpha \in [0{,}001;\ 0{,}01]$ та роблять підрахунок значень імовірностей P-value, що перевищують установлений рівень α для кожного з q тестів, тобто визначають коефіцієнт:

$$r_j = \frac{\#\{P_{ij} \geq \alpha,\ i = 1, 2, \ldots, m\}}{m}. \qquad (7.2)$$

У результаті формують вектор коефіцієнтів $R = \{r_1, r_2, \ldots, r_q\}$, елементи якого у відсотках характеризують проходження послідовності S_i усіх статистичних тестів.

Правило 1. Уважають, що генератор G пройшов тестування за j-м тестом, якщо значення коефіцієнта r_j перебуває в межах довірчого інтервалу $[r_{min}, r_{max}]$. Межі інтервалу визначають, відповідно до такого виразу:

$$r_{max(min)} = \hat{p} \pm 3\sqrt{\frac{\hat{p}(1 - \hat{p})}{m}}, \qquad (7.3)$$

де $\hat{p} = 1 - \alpha$.

5. Здійснюють аналіз статистичного портрету. Знайдені значення ймовірностей P_{ij} мають підкорятися рівномовірному закону розподілу на інтервалі [0, 1]. Для кожного вектора-стовпця статистичного портрету будують гістограму частоти F_k потрапляння значень P_{ij} у кожний із $k = 1, 2, \ldots, 10$ підінтервалів, на які розподілений інтервал [0, 1]. Рівномовірність розподілу значень імовірностей P_{ij} перевіряють із використанням критерію χ^2. Для цього розраховують статистику такого виду:

$$\chi_j^2 = \sum_{k=1}^{10} \frac{(F_k - m/10)^2}{m/10}, \qquad (7.4)$$

яка підкорюється розподілу χ^2 з дев'ятьма ступенями свободи.

Правило 2. Уважають, що генератор G пройшов тестування за j-м тестом, якщо виконується умова $P(\chi_j^2) > 0{,}0001$. Остаточне рішення ухвалюють, відповідно до правила:

"Уважають, що генератор G пройшов статистичне тестування пакетом NIST STS, якщо значення коефіцієнтів r_j для всіх $j = \overline{1,q}$ перебувають у межах довірчого інтервалу $[r_{min}, r_{max}]$ і виконується умова $P(\chi_j^2) > 0{,}0001$ для всіх $j = \overline{1,q}$."

7.6. Опис лабораторної установки

Як лабораторну установку використовують ПЕОМ з інстальованим програмним забезпеченням для генерування та тестування послідовностей випадкових і псевдовипадкових чисел.

Наявні засоби тестування NIST працюють під управлінням операційної системи Windows. Загальний вигляд вікна застосунку наведено на рис. 7.3.

Необхідно розглянути більш детально елементи управління застосунку. Є можливість протестувати вже наявну послідовність, що зберігають у файлі, вибравши залежний перемикач "Вхідний файл" (Input File), або сформувати нову за допомогою генераторів (потрібно вибрати один із залежних перемикачів у групі "Опції генератора" (Generator Options)) (рис. 7.4).

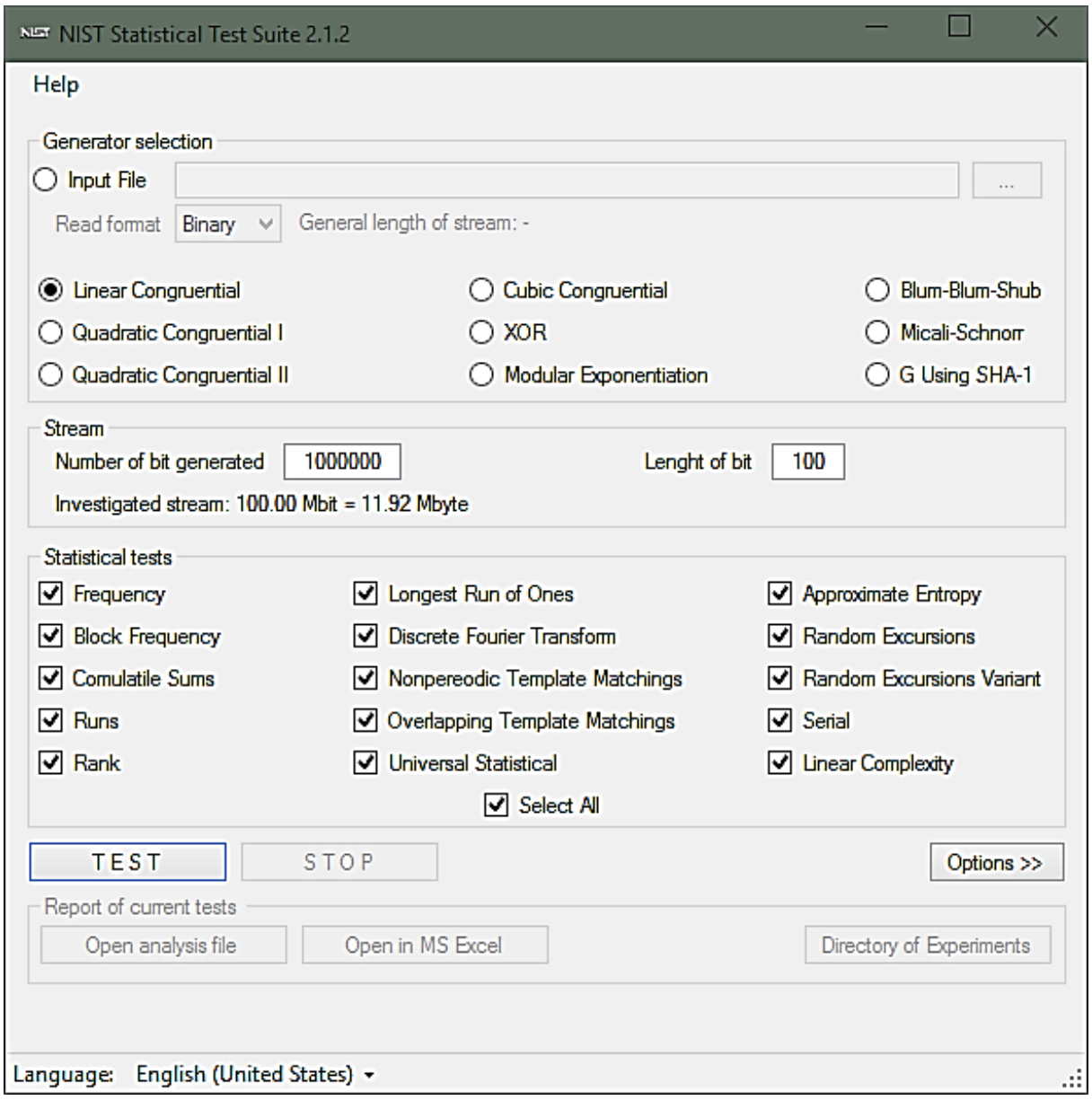


Рис. 7.3. **Загальний вигляд програми**

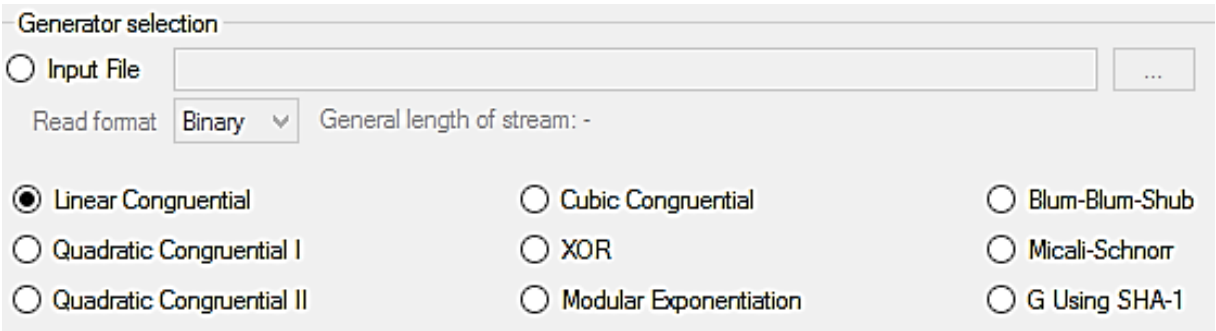


Рис. 7.4. **Робоче вікно застосунку**

У табл. 7.4 наведено перелік можливих генераторів.

Таблиця 7.4

Перелік можливих генераторів

№ п/п	Англійські назви	Українські назви
1	G using SHA-1	Генератор, що використовує SHA-1
2	Linear Congruential	Лінійний конгруентний генератор
3	Blum-Blum-Shub	Генератор Блума – Блума – Шуба
4	Micali-Schnorr	Генератор Мікалі – Шнорра
5	Quadratic Congruential I	Квадратичний конгруентний генератор (I варіант)
6	Quadratic Congruential II	Квадратичний конгруентний генератор (I варіант)
7	Cubic Congruential	Кубічний конгруентний генератор
8	XOR	Генератор, що застосовує XOR
9	ANSI X9.17 (3-DES)	Генератор на основі потрійного DES
10	G using DES	Генератор, що використовує DES
11	Modular Exponentiation	Степеневий модульний генератор
12	G from SHA-256	Генератор на основі SHA-256

Після вибору способу, за допомогою якого було визначено послідовність, необхідно вибрати, у якому вигляді буде подано цю послідовність бітів: в ASCII-форматі чи у бінарному форматі 16-річними розрядами. У ході лабораторної роботи необхідно вибирати бінарний формат.

Призначення наступного діалогового вікна – вибір тестів, які будуть застосовувати, і внесення їхніх параметрів (рис. 7.5).

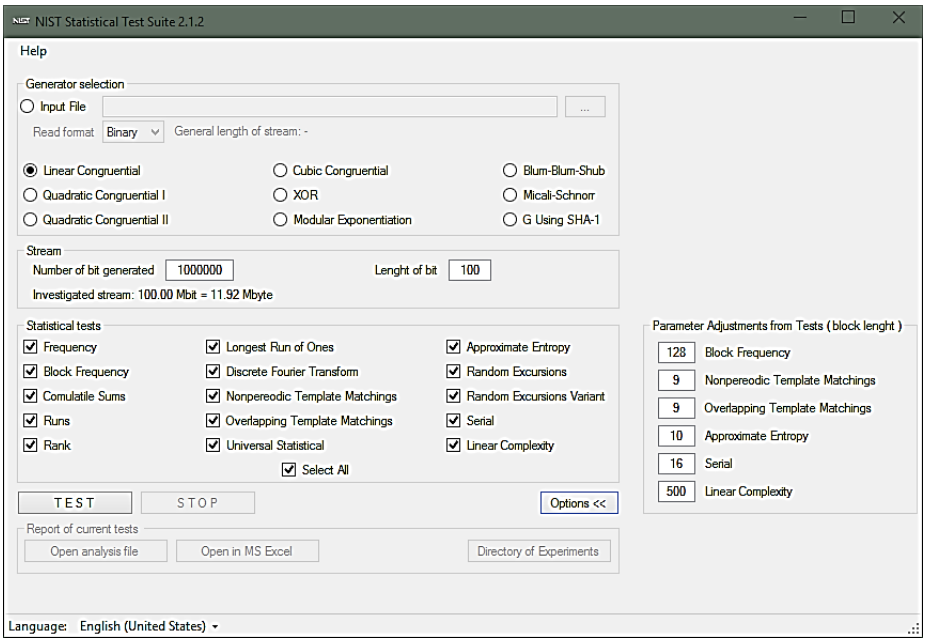


Рис. 7.5. **Вибір та визначення параметрів для тестів із параметрами**

Усього за методикою NIST STS випадкові та псевдовипадкові послідовності тестують на відповідність 16 тестам. Але, ураховуючи те, що деякі тести мають запускати з різними параметрами, загальна кількість тестів дорівнює 189. Усі тести можна умовно розподілити на тести з параметрами (Parameterized) і тести без параметрів (Non-parameterized). У засобі тестування ці тести розміщені на двох різних укладках. Тест уважають вибраним, якщо біля нього встановлено прапорець. Після вибору активного стану прапорця для тесту з параметрами необхідно ввести їхні значення (рис. 7.6), для тесту без параметрів достатньо лише активувати прапорець (рис. 7.7). За замовчуванням розробники програмного застосунку вже проставили параметри для тестів і довжини послідовності, які дорівнюють 1 млн бітів. Якщо довжина тестованої послідовності має інше значення, необхідно обчислити параметри за таблицею, наведеною в теоретичному матеріалі. Для прискорення вибору можливе використання двох кнопок: "Вибрати для тестування всі тести" або "Скасувати активність усіх тестів".

Примітка. Друга кнопка з'являється лише після натискання на першу.

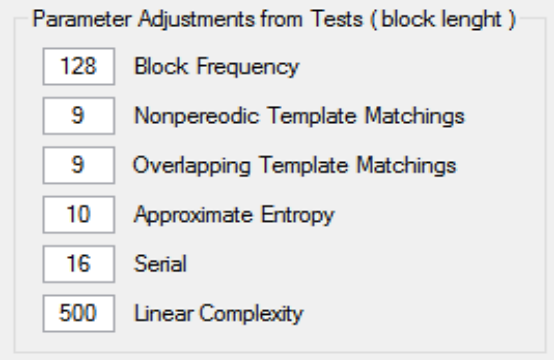


Рис. 7.6. **Визначення значень для тестів із параметрами**

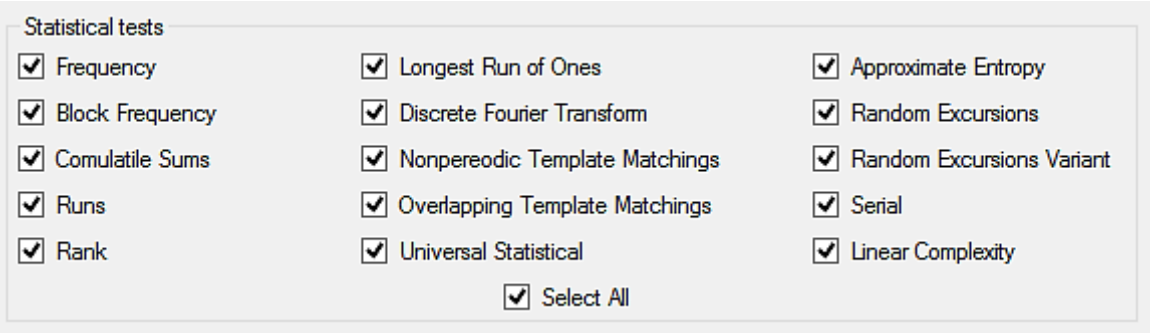


Рис. 7.7. **Вибір та визначення значень для тестів без параметрів**

Після вибору тестів необхідно у відповідних елементах управління визначити довжину послідовності (Length of bit stream) і кількість таких послідовностей (Number of bit streams generated). Натискання кнопки TEST викликає процес тестування (рис. 7.8).

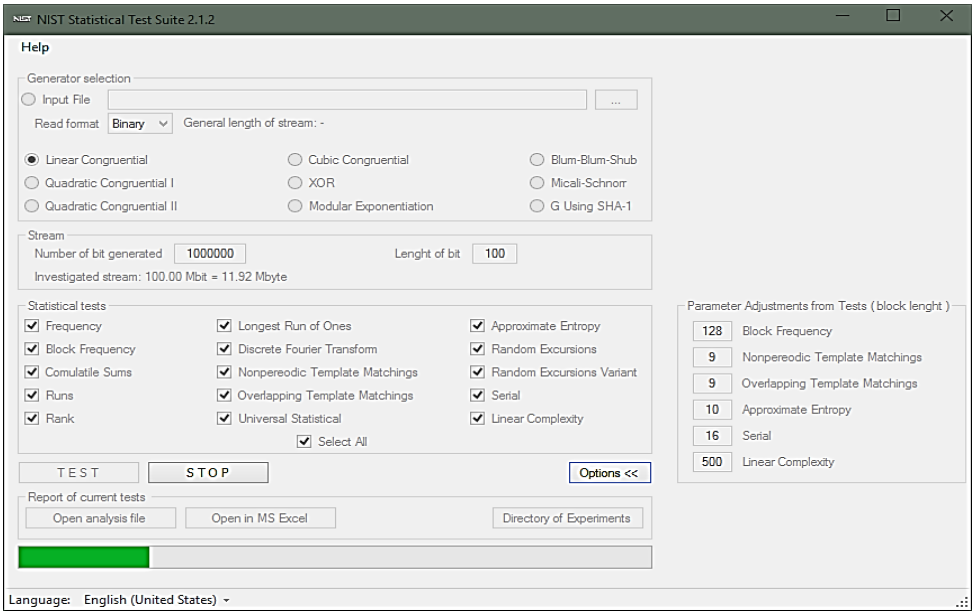
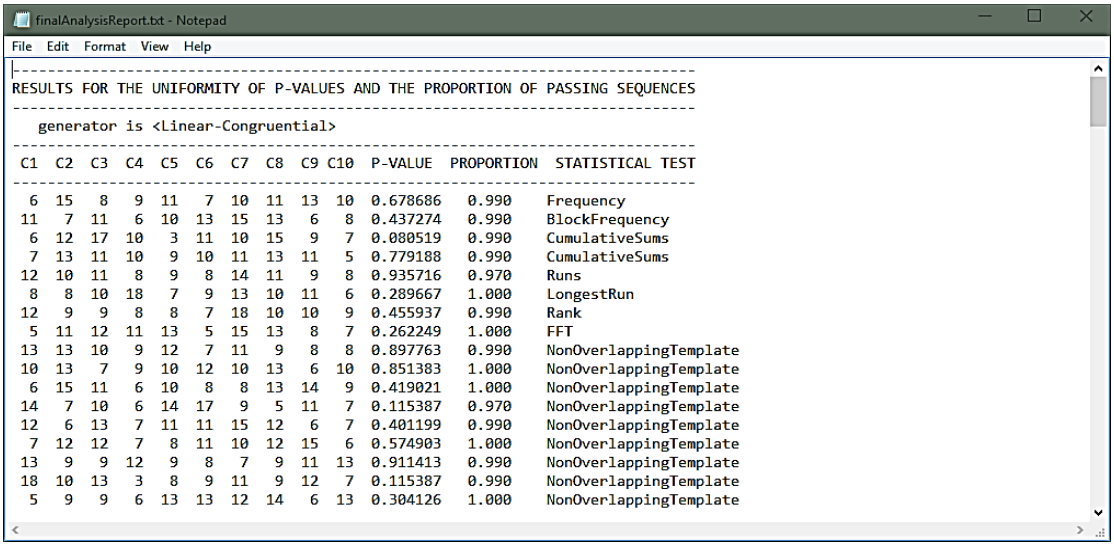


Рис. 7.8. **Меню налаштувань тестування**

Після закінчення роботи програми всі сумарні розрахункові дані розміщують у кореневому каталозі у файлі finalAnalysisReport. На рис. 7.9 показано типовий звіт програмного модуля щодо тестування псевдовипадкової послідовності.


finalAnalysisReport.txt - Notepad

File Edit Format View Help

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <Linear-Congruential>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  6  15   8   9  11   7  10  11  13  10  0.678686   0.990     Frequency
 11   7  11   6  10  13  15  13   6   8  0.437274   0.990     BlockFrequency
  6  12  17  10   3  11  10  15   9   7  0.080519   0.990     CumulativeSums
  7  13  11  10   9  10  11  13  11   5  0.779188   0.990     CumulativeSums
 12  10  11   8   9   8  14  11   9   8  0.935716   0.970     Runs
  8   8  10  18   7   9  13  10  11   6  0.289667   1.000     LongestRun
 12   9   9   8   8   7  18  10  10   9  0.455937   0.990     Rank
  5  11  12  11  13   5  15  13   8   7  0.262249   1.000     FFT
 13  13  10   9  12   7  11   9   8   8  0.897763   0.990     NonOverlappingTemplate
 10  13   7   9  10  12  10  13   6  10  0.851383   1.000     NonOverlappingTemplate
  6  15  11   6  10   8   8  13  14   9  0.419021   1.000     NonOverlappingTemplate
 14   7  10   6  14  17   9   5  11   7  0.115387   0.970     NonOverlappingTemplate
 12   6  13   7  11  11  15  12   6   7  0.401199   0.990     NonOverlappingTemplate
  7  12  12   7   8  11  10  12  15   6  0.574903   1.000     NonOverlappingTemplate
 13   9   9  12   9   8   7   9  11  13  0.911413   0.990     NonOverlappingTemplate
 18  10  13   3   8   9  11   9  12   7  0.115387   0.990     NonOverlappingTemplate
  5   9   9   6  13  13  12  14   6  13  0.304126   1.000     NonOverlappingTemplate
```

Рис. 7.9. **Звіт програмного модуля**

Якщо послідовність не пройшла тестування біля назви тесту з'являється зірочка. Для більш детального аналізу послідовного проходження кожного тесту можна використати файли, розміщені в поточному каталозі \experiments\AlgorithmTesting\Назва тесту.

7.7. Завдання на дослідження

Проведіть статистичні випробування визначених генераторів випадкових та псевдовипадкових послідовностей за методикою NIST STS. Зробіть аналіз визначених результатів та сформуйте акт випробувань.

У ході виконання лабораторної роботи необхідно:

1) за допомогою спеціального програмного забезпечення сформувати псевдовипадкову послідовність;

2) здійснити тестування з використанням лабораторної установки;

3) зробити аналіз результатів шляхом застосування статистичних критеріїв χ^2 Пірсона та Колмогорова – Смирнова:

- побудувати гістограму розподілу частот, виходячи з визначених результатів, та порівняти її з теоретичним розподілом, використовуючи критерій χ^2;
- побудувати емпіричну функцію розподілу, порівняти її з теоретичним розподілом, використовуючи критерій Колмогорова;
- заповнити таблицю результатів статистичних досліджень, у якій порівняти результати з еталонною вибіркою BBS;

4) підготувати та оформити звіт із виконаної лабораторної роботи.

Під час підготовки до роботи студент має підготувати бланк звіту з лабораторної роботи. Для цього необхідно:

- записати у звіті назву та мету роботи;
- скласти та занести у звіт програму експериментальних досліджень;
- дати стислу характеристику апаратного та програмного забезпечення, їхнього призначення, можливості та порядок використання;
- підготувати, за потреби, таблиці та графіки;
- унести до звіту основні аналітичні співвідношення.

Порядок виконання роботи

1. Запустіть застосунок статистичного тестування NIST_STS.exe.
2. Виконайте статистичні випробування вказаних викладачем ГВП.

Для цього використайте опис програмного забезпечення, наведеного в цих указівках.

3. Із використанням загальних результатів тестування, які містяться у файлі *finalAnalysisReport*, зробіть інтерпретацію результатів випробувань (див. приклад у додатку Б).

4. Із використанням результатів тестування, які містяться у файлі *finalAnalysisReport*, для вказаних викладачем статистичних тестів побудуйте теоретичну й емпіричну гістограми частоти F_k потрапляння значень P_{ij} у кожний із $k = 1, 2, \ldots, 10$ підінтервалів, на які розподілено інтервал [0, 1] (рис. 7.10).

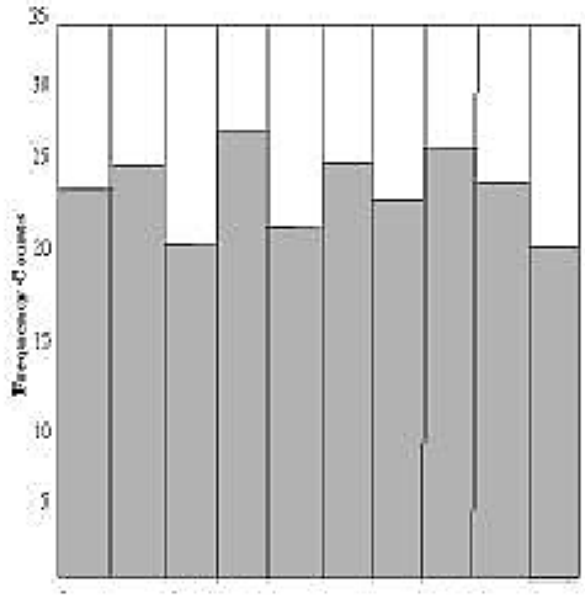


Рис. 7.10. **Зразок побудови гістограми**

Відповідно до виразу (7.4) розрахуйте значення χ^2 і перевірте погодженість теоретичного й емпіричного законів розподілу значень P_{ij} з $\alpha = 0{,}01$ і $\alpha = 0{,}05$. Критичні значення для χ^2 наведено в додатку В.

5. Перевірте погодженість емпіричного й теоретичного законів розподілу значень P_{ij} для вказаних викладачем статистичних тестів за допомогою критерію Колмогорова – Смирнова. Для цього з використанням результатів тестування, що містяться у файлі result відповідного статистичного тесту, побудуйте емпіричну функцію розподілу і теоретичну функцію розподілу для рівномовірного закону. Обчисліть статистику Колмогорова – Смирнова. Як критерій оцінювання розбіжності теоретичної $F_m(P_{ij})$

та емпіричної $F_e(P_{ij})$ функцій розподілу, розглядають максимальне значення модуля різниці:

$$D = max\left|F_T(P_{ij}) - F_E(P_{ij})\right|. \quad (7.5)$$

6. Обчисліть значення:

$$\beta = D\sqrt{m}, \quad (7.6)$$

де m – кількість точок, за якими будували емпіричну криву.

7. Знаючи β, за таблицею можна знайти ймовірність $P(\beta)$. Якщо ця ймовірність мала, то гіпотезу про погодженість законів відхиляють, і навпаки, якщо ймовірність велика, то гіпотезу приймають.

8. Перевірте погодженість результатів, які були визначені за допомогою критерію χ^2 і критерію Колмогорова – Смирнова, і зробіть висновки.

Примітка. Із метою спрощення виконання пунктів 7, 8, необхідно використовувати електронні таблиці Microsoft Excel.

7.8. Контрольні запитання

1. Які режими тестування використовує NIST?
2. Що є метою комплексного контролю?
3. У чому полягає статистика частотного (монобітного) тесту?
4. Скільки тестів містяться в пакеті NIST?
5. Скільки значень імовірності фактично обчислюють у пакеті NIST?
6. Що є статистичним портретом ГВП?
7. Яка найбільш поширена методика тестування криптографічних засобів захисту інформації?
8. Що є метою оперативного контролю?
9. У чому полягає статистика тесту перевірки випадкових відхилень?
10. Який підхід до визначення псевдовипадковості найбільш поширено на практиці?
11. Який дефект виявляють у результаті універсального тесту Маурера та частотного (монобітного) тесту?

Рекомендована література

1. Горбенко І. Д. Прикладна криптологія: теорія, практика, застосування // О. І. Горбенко, І. Д. Горбенко. – Харків : Форт, 2012. – 880 с.
2. Кузнецов О. О. Захист інформації в інформаційних системах. Методи традиційної криптографії : навч. посіб. / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Харків : Вид-во ХНЕУ, 2010. – 316 с.
3. Кузнецов О. О. Захист інформації та економічна безпека підприємства : монографія / О. О. Кузнецов, С. П. Євсеєв, С. В. Кавун. – Харків : Вид-во ХНЕУ, 2008. – 360 с.
4. Лукацкий А. Обнаружение атак / А. Лукацкий. – Санкт-Петербург : BHV-Петербург, 2001. – 624 с.
5. Масленников М. Е. Практическая криптографія / М. Е. Масленников. – Санкт-Петербург : BHV, 2003. – 458 с.
6. Милославская Н. Р. Интрасети: доступ в Интернет, защита / Н. Р. Милославская, А. И. Толстой. – Москва : Юнити-Дана, 2000. – 527 с.
7. Молдовян А. А. Криптография / А. А. Молдовян, Н. А. Молдовян, Б. Я. Советов. – Санкт-Петербург : Лань, 2000. – 224 с. – (Серия "Учебники для вузов. Специальная литература").
8. НД ТЗІ 2.5-005-99. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу. – Київ : ДСТСЗІ СБУ, 1999 р. – 18 с.
9. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації у комп'ютерних системах від несанкціонованого доступу. – Київ : ДСТСЗІ СБУ 1999 р. – 67 с.
10. НД ТЗІ 3.7-001-99. Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі. – Київ : ДСТСЗІ СБУ 1999 р. – 35 с.
11. НД ТЗІ 1.1-003-99. термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу. – Київ : ДСТСЗІ СБУ 1999 р. – 22 с.
12. Основи захисту інформації : навч. посіб. / О. А. Смірнов, Л. Г. Віхрова, С. І. Осадчий та ін. – Кіровоград: КНТУ, 2010. – 322 с.
13. Основы информационной безопасности : учеб. пособ. для вузов / Е. Б. Белов, В. П. Лось, Р. В. Мещеряков, А. А. Шелупанов. – Москва : Горячая линия – Телеком, 2006. – 544 с.

14. Остапов С. Е. Основи криптографії / С. Е. Остапов, Л. О. Валь. – Чернівці : Книги XXI, 2008. – 188 с.

15. Остапов С. Е. Технології захисту інформації / С. Е. Остапов, С. П. Євсєєв, О. Г. Король. – Чернівці : Чернівецький національний університет, 2013. – 471 с.

16. Поповский В. В. Защита информации в телекоммуникационных системах / В. В. Поповский, А. В. Персиков. – Харьков : ООО "Компания СМИТ", 2006. – Т.1. – 292 с.

17. Потий А. В. Стандартизация и сертификация в сфере защиты информации. Стандарты механизмов безопасности : учеб. пособ. / А. В. Потий. – Харьков : ХНУРЭ, 2002. – 80 с.

18. Сингх С. Книга шифров / С. Сингх. – Москва : АСТ: Астрель, 2007. – 447 с.

19. Столингс В. Криптография и защита сетей / В. Столингс. – Москва : Вильямс, 2004. – 848 с.

20. Таненбаум Э. Современные операционные системы / Э. Таненбаум. – Санкт-Питербург. – 2010. – 1120 с.

21. Трубачев А. П. Оценка безопасности информационных технологий / под общ. ред. В. А. Галатенко. – Москва : СИП РИА, 2001. – 356 с.

22. Хорошко В. А. Методы и средства защиты информации / В. А. Хорошко, А. А. Чекатков. – Киев : Юниор, 2003 – 504 с.

23. Чмора А. Л. Современная прикладная криптография / А. Л. Чмора. – Москва : Гелиос АРВ, 2001. – 256 с.

24. Шеннон К. Э. Теория связи в секретных системах / К. Э. Шеннон. Работы по теории информации и кибернетике. – Москва : ИЛ, 1963. – С. 333–402.

25. Шеховцов В. А. Операційні системи / В. А. Шеховцов. – Санкт-Петербург : BHV, 2006. – 576 с.

26. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С / Б. Шнайер. – 2-е изд. – Москва : Гелиос, 2001. – 308 с.

27. Щеглов А. Ю. Защита компьютерной информации от несанкционированного доступа / А. Ю. Щеглов. – Санкт-Петербург : Наука и Техника, 2004. – 384 с.

28. Алгоритм НОТАРИУС-1. [Электронный ресурс]. – Режим доступа : http://www.lancrypto.com/products/index.php?ID=1068.

29. Концепція технічного захисту інформації в Україні [Електронний ресурс]. – Режим доступу : http://zakon1.rada.gov.ua/cgibin/laws/main.cgi?nreg=1126-97-%EF.

30. Положення про проведення відкритого конкурсу криптографічних алгоритмів : ДСТСЗІ / Інститут кібернетики ім. В. М. Глушкова НАНУ. – [Електронний ресурс]. – Режим доступу : http://www.dstszi.gov.ua/dstszi/control/ru/publish/article;jsessionid=EE63A37FEF8F5B34030F1E%2038D7247DBC?art_id=48387&cat_id=92733.

31. Про державну таємницю [Електронний ресурс] : Закон України № 3855-12 від 21.01.1994 р. – Режим доступу : http://zakon.rada.gov.ua/cgibin/laws/main.cgi?nreg=3855-12.

32. Про захист інформації в інформаційно-телекомунікаційних системах [Електронний ресурс] : Закон України № 80/94-ВР від 05.07.1994 р. – Режим доступу : http://www.ucrf.gov.ua/uk/doc/laws/1147239096.

33. Про інформацію [Електронний ресурс] : Закон України № 2657-XII від 21.12.2019 р. – Режим доступу : http://www.libr.dp.ua/misc/law_04.html.

34. Ростовцев А. Г. Методы криптоанализа классических шифров [Электронный ресурс] / А. Г. Ростовцев, Н. В. Михайлова. – Режим доступа : http://crypto.hotbox.ru.

35. Украинский ресурс по безопасности [Электронный ресурс]. – Режим доступа : http://kiev-security.org.ua.

36. Brown D. R. L. Generic Groups, Collision Resistance, and ECDSA [Electronic resource] / D. R. L. Brown. – Access mode : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.1093.

37. NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition [Electronic resource]. – Access mode : http://www.nist.gov/itl/csd/sha-100212.cfm.

Додатки

Додаток А

Побудова емпіричної функції розподілу та функції візуалізації результатів

Методика побудови емпіричної функції розподілу із використанням електронних таблиць Microsoft Excel:

1) відкрийте файл results у каталозі \experiments\AlgorithmTesting\ Назва тесту;

2) скопіюйте значення, що перебувають в ньому;

3) уставте значення в електрону книгу Microsoft Excel;

4) приведіть значення у відповідність до формату реальних чисел, узятих у Microsoft Excel;

5) виконайте упорядкування значень;

6) проставте в сусідньому стовпчику значення теоретичної функції розподілу нормального закону;

7) знайдіть максимальну різницю між теоретичними та емпіричним значеннями, зробити відповідні висновки за критерієм Колмогорова;

8) побудуйте графіки теоретичної та емпіричної функції розподілу нормального закону засобами Microsoft Excel.

Додаток Б

Результати статистичного тестування програмної реалізації алгоритму блокового шифрування FIPS 197

Випробування проводили представники на випробувальному стенді у спеціальній лабораторії АТ "Інститут інформаційних технологій", відповідно до методики тестування NIST SP 800-22.

Інтерпретація результатів перевірки програмної реалізації алгоритму блокового шифрування FIPS-197 у режимі лічильника полягає в такому. Програмну реалізацію алгоритму блокового шифрування FIPS-197 у режимі лічильника піддавали статистичному тестуванню з використанням методики NIST STS, рекомендованої Національним інститутом стандартизації та технологіями США – NIST SP 800-22.

Із використанням методики NIST STS було здійснено тестування програмної реалізації алгоритму блокового шифрування FIPS-197 у режимі лічильника, із метою порівняльного аналізу генератора псевдовипадкових чисел BBS (тестову вибірку, рекомендовану NIST).

Для здійснення тестувань було вибрано такі параметри:

- довжину послідовності, що тестують, – $n = 10^6$ бітів;
- кількість послідовностей, що тестують, – $m = 100$;
- рівень значущості $\alpha = 0{,}01$. Таким чином, обсяг вибірки, що тестують, становить $n = 10^6 \times 100 = 10^8$ бітів;
- кількість тестів (q) для різних довжин $q = 189$. Таким чином, статистичний портрет генератора містить 18 900 значень імовірності P.

В ідеальному випадку за $m = 100$ і $\alpha = 0{,}01$ у ході тестування може бути відкинуто тільки одну послідовність зі 100, тобто коефіцієнт проходження кожного тесту має становити 99 %. Але це занадто жорстке правило. Тому застосовують правило на основі довірчого інтервалу для r_j. Нижня межа в цьому разі становить значення $r_{min} = 0{,}96015$. Із цих позицій аналізують результати тестування генераторів.

У табл. Б.1 наведено результати проходження тестування програмної реалізації алгоритму блокового шифрування FIPS 197 у режимі лічильника за правилом 1.

Таблиця Б.1

Результати проходження тестування програмної реалізації алгоритму блокового шифрування FIPS 197 за правилом 1

Генератори	Кількість тестів, у яких тестування пройшли понад 99 % послідовностей	Кількість тестів, у яких тестування пройшли понад 96 % послідовностей
BBS	134 (71 %)	189 (100 %)
FIPS 197	126 (67 %)	189 (100 %)

Усі тести пройшли (або вказати який тест не пройшов та, відповідно до керівництва NIST STS, надати інтерпретацію недоліку, виявленого тестом).

У табл. Б.2 наведено результати проходження тестування програмної реалізації алгоритму блокового шифрування FIPS-197 у режимі лічильника за правилом 2.

Таблиця Б.2

Результати проходження тестування програмної реалізації алгоритму блокового шифрування FIPS 197 за правилом 2

Генератори	Кількість тестів, у яких значення $P < 0,001$	Кількість тестів, у яких значення $P < 0,01$
BBS	134 (71 %)	189 (100 %)
FIPS 197	126 (67 %)	189 (100 %)

Якщо необхідно вказати за якими тестами значення P було нижчим від визначеної межі, укажіть, які недоліки виявляють ці тести; порівняйте ці результати з результатами тестування за правилом 1.

У додатку В наведено результати обчислення, а також побудовано статистичний портрет.

Таким чином, можна зробити висновок, що програмна реалізація алгоритму блокового шифрування FIPS-197 у режимі лічильника пройшла оперативний контроль за методикою FIPS 140-2 і комплексний контроль за методикою NIST STS.

Критерії погодженості

У побудові ключових систем одним з основних завдань є визначення випадкових і псевдовипадкових послідовностей, які не відрізняються від випадкових та мають великий період. Після генерації послідовності чисел $X = \{x_1, x_2, \ldots, x_n\}$ необхідно впевнитися в тому, що випадкова величина X має рівномірний закон розподілу, її реалізації випадкові та незалежні. Математична статистика дає нам можливість побудувати статистичні тести для перевірки гіпотез про рівномірність, випадковість і незалежність випадкових величин.

Для перевірки гіпотези про закон розподілу слід скористатися критеріями χ^2 Пірсона та Колмогорова – Смирнова.

Критерій Пірсона χ^2 перевіряє погодженість гіпотетичних імовірностей $P_k = P(x_k)$ випадкових величин x_1, x_2, …, x_n із їхніми відносними частотами $h_k = v_k / n$ у вибірці з n незалежних спостережень. Статистика критерію має такий вигляд:

$$\chi^2 = n \sum_{k=1}^{m} \frac{(h_k - p_k)^2}{p_k},$$

де m – кількість інтервалів розподілу.

Граничне значення статистики для рівня значущості визначають за такою формулою:

$$\chi^2_\alpha \approx l \left(1 - \frac{2}{9l} + z_\alpha \sqrt{\frac{2}{9l}} \right)^3,$$

де l – кількість ступенів свободи;

z_α – граничне значення стандартного нормального розподілу 7.

Гіпотезу про погодженість емпіричного закону розподілу спостережуваної випадкової величини з теоретичним відкидають, якщо спостережене значення $\chi^2 > \chi^2_\alpha$ (табл. В.1).

Таблиця В.1

Критичні значення χ^2-статистики

Ступені свободи	*p* = 99 %	*p* = 95 %	*p* = 75 %	*p* = 50 %	*p* = 25 %	*p* = 5 %	*p* = 1 %
1	0,00016	0,00393	0,1015	0,4549	1,323	3,841	6,635
2	0,00201	0,1026	0,5753	1,386	2,773	5,991	9,210
3	0,1148	0,3518	1,213	2,366	4,108	7,815	11,34
4	0,2971	0,7107	1,923	3,357	5,385	9,488	13,28
5	0,5543	1,1455	2,675	4,351	6,626	11,07	15,09
6	0,8720	1,635	3,455	5,348	7,841	12,59	16,81
7	1,239	2,167	4,225	6,346	9,037	14,07	18,48
8	1,646	2,733	5,071	7,344	10,22	15,51	20,09
9	2,088	3,325	5,889	8,343	11,39	16,92	21,67
10	2,558	3,940	6,737	9,342	12,55	18,31	23,21
11	3,053	4,575	7,584	10,34	13,70	19,68	24,73
12	3,571	5,226	8,438	11,34	14,84	21,03	26,22
15	5,229	7,261	11,04	14,34	18,25	25,00	30,58
20	8,260	10,85	15,45	19,34	23,85	31,41	37,57
30	14,95	18,49	24,48	29,34	34,80	43,77	50,89
50	29,71	34,76	42,94	49,33	56,33	67,50	76,15
> 30	Приблизно $v + 2\sqrt{v \times x_p} + \frac{4}{3} \times \frac{2}{p} - \frac{2}{3}$						
x_p	-2,33	-1,64	-0,675	0,00	0,675	1,64	2,33

Критерій Колмогорова – **Смирнова** засновано на розподілі такої величини:

$$D_n = max\ |F_n(x) - F(x)|,$$

де $F_n(x)$ – емпіричний закон розподілу;

$F(x)$ – гіпотетичний закон розподілу.

Відомо, що яка б не була неперервна функція розподілу $F(x)$, імовірність $P\left\{D_n < \frac{\lambda}{\sqrt{n}}\right\}$ за $n \to \infty$ прямує до межі $K(\lambda) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1} e^{-2\lambda^2 k^2}$.

Для практичних обчислень варто застосовувати такі формули:

$$D_n^+ = \max_{1 \le m \le n} \left(\frac{m}{n} - F(\xi_m) \right);$$

$$D_n^- = \max_{1 \le m \le n} \left(F(\xi_m) - \frac{m-1}{n} \right);$$

$$D_n = max\left(D_n^+, D_n^-\right).$$

де $\xi_1 \le \xi_2 \le \ldots \le \xi_n$ – упорядковані значення випадкової величини.

Статистика D_n має певний розподіл (розподіл Колмогорова (табл. В.2)), протабульований для деяких значень n. За $n \ge 10$ для визначення граничного значення $D_n(\alpha)$ на відрізку $0{,}01 \le \alpha \le 0{,}2$ варто користуватися такою формулою:

$$D_n(\alpha) = \sqrt{\frac{1}{2n}(y) - \frac{2y^2 - 4y - 1}{18n}} - \frac{1}{6n} \approx \sqrt{\frac{y}{2n}} - \frac{1}{6n},\ y = -ln(0{,}5\alpha).$$

За $n \ge 100$ зазначена формула правильна для всіх $0{,}0001 \le \alpha \le 0{,}5$.

Якщо в результаті досліду виявиться, що $D_n \ge D_n(\alpha)$, то гіпотезу про погодженість емпіричного й гіпотетичного законів розподілу варто відкинути з рівнем значущості α.

Таблиця В.2

Дані для критерію Колмогорова – Смирнова

β	$P(\beta)$	β	$P(\beta)$	β	$P(\beta)$	β	$P(\beta)$	β	$P(\beta)$	β	$P(\beta)$
0	1,0	0,3	1,0	0,6	0,654	0,9	0,393	1,2	0,112	1,5	0,022
0,1	1,0	0,4	0,997	0,7	0,711	1,0	0,270	1,3	0,068	1,6	0,012
0,2	1,0	0,5	0,994	0,8	0,544	1,1	0,179	1,4	0,040	1,7	0,006

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

Євсеєв Сергій Петрович
Мілов Олександр Володимирович
Король Ольга Григорівна

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ОСНОВ КРИПТОГРАФІЧНОГО ЗАХИСТУ

Навчальний посібник

Самостійне електронне текстове мережеве видання

Відповідальний за видання *С. П. Євсеєв*

Відповідальний редактор *М. М. Оленич*

Редактор *О. Г. Доценко*

Коректор *О. Г. Доценко*

План 2020 р. Поз. № 39-ЕНП. Обсяг 222 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру
ДК № 4853 від 20.02.2015 р.