Ю. Є. Яремчук, О. В. Салієва, І. О. Бондаренко

ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

Міністерство освіти і науки України
Вінницький національний технічний університет

Ю. Є. Яремчук, О. В. Салієва, І. О. Бондаренко

ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

**Електронний навчальний посібник
комбінованого (локального та мережного) використання**

Вінниця
ВНТУ
2024

УДК 004.056.55(075.8)
Я72

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 6 від 28.11.2024 р.)

Рецензенти:
С. В. Лазаренко, доктор технічних наук, професор
А. П. Саміла, доктор технічних наук, професор
О. В. Бісікало, доктор технічних наук, професор
О. П. Войтович, кандидат технічних наук, доцент

Яремчук, Ю. Є.

Я72 Основи криптографічного захисту інформації : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / Яремчук Ю. Є., Салієва О. В., Бондаренко І. О. – Вінниця : ВНТУ, 2024. – 139 с.

Посібник присвячений матеріалам лекційного курсу з дисципліни «Основи криптографічного захисту інформації» для здобувачів вищої освіти, що навчаються за спеціальністю 125 «Кібербезпека та захист інформації» (освітньо-професійні програми «Управління інформаційною безпекою» та «Кібербезпека інформаційних технологій та систем»).

Мета посібника – надати здобувачам вищої освіти можливість більш детально вивчити аудиторний матеріал, опрацювати теми відведені на лабораторні заняття і підготуватися до підсумкового контролю знань, а також застосувати отримані знання для подальшої фахової роботи в галузі інформаційної безпеки.

УДК 004.056.55(075.8)

ЗМІСТ

ВСТУП

Бурхливий розвиток інформаційних технологій сприяв стрімкому зростанню обсягів цифрових даних, які передаються по відкритих каналах зв'язку. Це, так само, створило такі нові проблеми, як кіберзлочинність, кібербулінг, збільшилася кількість кібератак і загроз для інформаційної безпеки.

У зв'язку з цим вагоме місце посідає вивчення основних криптографічних методів захисту інформації, які дозволяють захищати дані від несанкціонованого доступу, шифруючи інформацію таким чином, що лише авторизовані користувачі можуть її прочитати. Крім того, використання криптографічних алгоритмів забезпечує цілісність даних, запобігаючи їх зміні або підробці. Це дозволяє виявляти будь-які спроби модифікації даних у разі їх передачі або зберігання. Також криптографічні методи забезпечують автентифікацію для підтвердження особи користувача або джерела даних. Це критично важливо для безпечної комунікації, електронної комерції, банківських операцій та інших галузей суспільної діяльності.

У запропонованому навчальному посібнику розглядаються принципи та методи створення надійних систем криптографічного захисту інформації, які можна використовувати на практиці.

Викладені принципи побудови криптосистем різного типу, починаючи від класичних до найсучасніших, з урахуванням їхніх сильних і слабких сторін. Зокрема, проаналізовано шифри заміни, перестановки, гамування. Розглянуто відомі методи їх криптоаналізу, зокрема на основі статистичного аналізу біграм, використання тесту Казіскі, визначення взаємного індексу збігу.

Наводяться описи і основні характеристики блокових шифрів: DES, Калина, ГОСТ 28147-2009, Rijndael та асиметричних криптоалгоритмів: RSA, Діффі-Хеллмана, Ель-Гамаля. Розглядаються принципи організації, функціонування та забезпечення надійності інфраструктури ключів.

Посібник призначений для здобувачів вищої освіти старших курсів бакалаврату зі спеціальності 125 «Кібербезпека та захист інформації», а також фахівців, які займаються впровадженням засобів криптографічного захисту інформації.

Практичні аспекти, відображені в посібнику, сприятимуть кращому розумінню та засвоєнню матеріалу здобувачами вищої освіти, розвитку у них навичок практичного застосування криптографічних методів.

Завдяки контрольним питанням, які наведено до кожного розділу, здобувачі вищої освіти можуть самостійно перевірити рівень засвоєння та закріплення теоретичного матеріалу.

Таким чином, цей посібник охоплює теоретичні аспекти та практичну реалізацію криптографічних методів захисту інформації, надає необхідні знання та інструменти для вирішення актуальних задач криптографії.

РОЗДІЛ 1 КЛАСИЧНІ МЕТОДИ ШИФРУВАННЯ

Криптографія – це наука, яка застосовує математичні методи для гарантування конфіденційності, цілісності та автентичності, тобто забезпечення захисту інформації від несанкціонованого доступу та модифікації з боку неавторизованих користувачів або процесів, а також підтвердження того, що суб'єкт або ресурс відповідають заявленій ідентичності.

Забезпечення конфіденційності даних здійснюється шляхом застосування криптографічних методів шифрування, а підтвердження їхньої цілісності та автентичності – за допомогою механізму цифрового підписування [1].

Класична криптографія досліджує різноманітні методи шифрування інформації, які використовувалися для захисту повідомлень у різні історичні епохи. Відомими шифрами класичної криптографії є шифри Цезаря, Віженера, Беласо та ін. Ці методи мали різні ступені складності та стійкості до криптоаналізу, що визначало їхню ефективність у захисті інформації.

1.1 Шифри заміни

Шифр заміни – це підстановка, яка встановлює взаємно однозначну відповідність між символами відкритого t_i та шифрованого текстів $s_i = X(t_i)$. Множина всіх таких підстановок над Z_m позначається $S(Z_m)$.

Група $S(Z_m)$ відносно операції добутку має такі властивості [1]:

1. Замкнутість: добуток двох підстановок X_1 і X_2 є підстановкою: $X_1X_2 \in S(Z_m)$. Дана властивість в термінах шифру заміни означає, що двократне застосування простої заміни еквівалентно деякій третій заміні. Тобто, стійкість шифрування від цього не збільшується.

2. Асоціативність: результат добутку $X_1X_2X_3$ не залежить від порядку розставляння дужок.

3. Існування одиниці: підстановка, визначена як $E = \binom{i}{i}$, є одиницею $S(Z_m)$ відносно операції множення: $\forall x \in S(Z_m) EX = XE = X$.

4. Існування оберненого елемента: для будь-якої підстановки X існує єдина обернена підстановка X^{-1}, яка задовольняє умову $XX^{-1} = X^{-1}X = E$.

Взаємно оберненими підстановками є:

$$X = \begin{pmatrix} 0 & 1 & \dots & m-1 \\ x_0 & x_1 & \dots & x_{m-1} \end{pmatrix} \text{ і } X^{-1} = \begin{pmatrix} x_0 & x_1 & \dots & x_{m-1} \\ 1 & 2 & \dots & m-1 \end{pmatrix}.$$

Ключі шифрування та розшифрування в даному алгоритмі є насправді елементами симетричної групи підстановок $S(Z_m)$, а їх загальна кількість становить $m!$.

1.1.1 Шифри простої моноалфавітної заміни

Найпростішим криптографічним шифром, який використовується для захисту відкритих текстів, є шифр простої моноалфавітної заміни, при якому кожен символ вихідного тексту, відповідно до встановленого прави-

ла, однозначно замінюється шифрованим символом.

У спрощеній версії відбувається заміна однієї букви на іншу, причому вхідний і вихідний алфавіти ідентичні за складом, але відрізняються перестановкою символів. Для зашифрування будь-якої букви відкритого тексту визначається її позиція у вхідному алфавіті, а на відповідне місце у шифротексті ставиться буква з тією ж позицією, але з вихідного алфавіту [2].

Наприклад, нехай вхідний та вихідний алфавіти мають, відповідно, вигляд:

А Б В Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я

К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я А Б В Г Ґ Д Е Є Ж З И І Ї Й

Тоді, маючи вхідне повідомлення: І_ЧУЖОМУ_НАУЧАЙТЕСЬ_Й_СВОГО_НЕ_ЦУРАЙТЕСЬ, отримаємо такий шифротекст: ХЖҐТЯЬҐЮКҐЖКЧГРВІЧВМЯНЯЮРЄҐБКЧГРВІ.

Математично даний алгоритм шифрування можна записати у вигляді

$$x_i \equiv a_i + k(mod 33), \qquad (1.1)$$

де x_i, a_i – букви, відповідно, шифрованого і відкритого текстів;

k – ключ шифрування (число, що задає величину циклічного зсуву алфавіту, у вищезазначеному прикладі $k = 15$).

Існують шифри двозначної заміни, де кожна буква вхідного алфавіту взаємно однозначно замінюється на пару цифр. Для зручності використання можуть застосовуватися відповідні таблиці, наприклад, табл. 1.1.

Таблиця 1.1 – Ключ шифру двозначної заміни

	0	1	2	3	4	5
0	А	Д	И	К	Н	Є
1	З	Б	О	С	Ф	Ц
2	Л	Ї	В	Ч	Й	Ш
3	П	У	Я	Г	Х	Т
4	Е	Щ	Р	І	М	Ь
5	Ґ	Ю	Ж			

Таким чином, букву вхідного тексту при шифруванні замінюють парою цифр, що відповідає номеру рядка та номеру стовпця, на перетині яких вона розташована. Наприклад, буква Т розміщена на перетині третього рядка і п’ятого стовпця, тому вона замінюється парою 35.

Поширеним є метод гаслового шифрування, де вихідний алфавіт формується як функція від двох ключових параметрів – букви і слова, що складається з різних букв. У цьому випадку перший параметр – «буква» визначає позицію вписування другого ключового параметра у нижньому рядку підстановки, а відсутні в слові-ключі букви розташовуються за алфавітом. Наприклад, якщо ключовим параметром (гаслом) є слово СИМЕТРІЯ, то вихідна послідовність букв відносно вхідної матиме вигляд:

А Б В Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я
А Б В Г Ґ Д Є Ж З Ї Й К Л Н О П **С И М Е Т Р І Я** У Ф Х Ц Ч Ш Щ Ь Ю

Для підвищення стійкості гаслових шифрів використовують різні прийоми побудови гаслових ключів простої заміни (табл. 1.2).

Таблиця 1.2 – Ключ шифру двозначної заміни з використанням гасла АЗИМУТ

	0	1	2	3	4	5
0	**А**	**З**	**И**	**М**	**У**	**Т**
1	Б	В	Г	Ґ	Д	Е
2	Є	Ж	І	Ї	Й	К
3	Л	Н	О	П	Р	С
4	Ф	Х	Ц	Ч	Ш	Щ
5	Ь	Ю	Я			

Також використовують шифри парної заміни з двозначним шифропозначенням. Проте створення таких таблиць для шифрування та розшифрування є досить складним завданням, адже таблиці є надзвичайно громіст-кими.

Приклад ключа шифру парної заміни з двозначним шифропозначенням відображено в табл. 1.3.

Таблиця 1.3 – Ключ шифру парної заміни з двозначним шифропозначенням

	Р	Б	К		Ю
А	БВ	ЮН	ЛЖ		РЬ
Б	ЦЮ	НГ	МК		АВ
В	ОЛ	ЗО	ПА		ЯР
…					
Я	ІЛ	ВЕ	ФЯ		ДВ

У цьому випадку процес шифрування відкритого тексту складається з послідовних кроків – тактів шифрування. Найменша одиниця відкритого тексту, що підлягає шифруванню протягом одного такту роботи шифру, називається елементом перетворення. На кожному такті шифрувальна система перетворює певну групу символів, з фіксованою кількістю знаків, що відображає значність групи. Так, групи з двох символів називаються біграмами, з трьох символів – три грамами тощо.

Основним недоліком шифрів простої однобуквеної заміни є те, що у шифрованому тексті зберігаються всі частотні характеристики відкритого тексту, всі сполучення і повторення. У зв’язку з цим, потрібно наблизити частоти появи букв до рівноймовірних. З цією метою кожній шифровеличині надається декілька шифропозначень, причому чим вища ймовірність

появи букви у відкритому тексті, тим більше шифрованих позначень їй призначається. Такий метод називається рандомізацією відкритого тексту.

Подібний ефект досягається й за допомогою «книжкових шифрів», де шифр складається з номера сторінки, рядка і позиції букви в рядку обраної книги, яку мають як відправник повідомлення, так і одержувач. Один з можливих варіантів книжкового шифру передбачає запис частини тексту книги в квадрат. При такому підході криптограма, зазвичай, містить посилання, яке вказує на координати початку тексту в книзі. Наведемо приклад можливого варіанта книжкового шифру (табл. 1.4).

Таблиця 1.4 – Книжковий шифр

	1	2	3	4	5	6	7	8	9	0
1	К	Л	А	С	М	Е	Т	О	Д	І
2	В	Ш	И	Ф	Р	У	В	А	Н	Н
3	Я	Я	К	І	З	В	О	Д	Я	Т
4	Ь	С	Я	Д	О	С	Т	В	О	Р
5	Е	Н	Н	Я	З	А	П	Е	В	Н
6	И	М	А	Л	Г	О	Р	И	Т	М
7	О	М	Т	А	Б	Л	И	Ц	І	Д
8	Е	Д	Л	Я	К	О	Ж	Н	О	Ї
9	Б	У	К	В	И	В	І	Д	К	Р
0	И	Т	О	Г	О	Т	Е	К	С	Т

Криптоаналіз шифру одноалфавітної заміни ґрунтується на аналізі частоти появи символів у зашифрованому тексті. Отриманий розподіл частот порівнюється з розподілом частот букв алфавіту вихідного повідомлення. За такої умови припускають, що буква з найбільшою частотою в шифротексті, ймовірно, відповідає букві з найбільшою частотою в мові оригіналу і под.

Таким чином, основними етапами криптоаналізу шифру одноалфавітної заміни на основі шифротексту є [3]:

1. Побудова діаграми розподілу частот шифротексту (у відсотковому співвідношенні) та букв алфавіту вихідного повідомлення;
2. Розташування частот у порядку зростання;
3. Знаходження можливого значення ключа як різниці між відповідними значеннями частот.

Потрібно зауважити, що ймовірність успішного розкриття системи шифрування підвищується зі збільшенням довжини шифротексту.

1.1.2 Шифри простої багатоалфавітної заміни

Для підвищення стійкості до криптографічних атак використовують шифри багатоалфавітної заміни, де для шифрування кожної букви використовують більше, ніж один алфавіт.

Наприклад, можна застосувати два алфавіти, один буде слугувати для

шифрування букв, що займають непарні позиції, а інший – парні. У цьому випадку, буква шифротексту матиме високу частоту тільки в тому випадку якщо її частота є високою в обох алфавітах. І, навпаки, буква шифротексту матиме низьку частоту, якщо вона «представляє» низькочастотні букви двох алфавітів. Отже, у шифротексті буде менше букв з дуже високою або низькою частотою, ніж у відкритому тексті. Побудувавши стовпчасту гістограму розподілу частот букв, можна побачити, що піки будуть нижчими, а западини – менш вираженими. Таким чином, застосування двох алфавітів вирівнює розподіл частот.

Це ж саме стосується і частоти біграм, оскільки будь-яка біграма, яка часто зустрічається в шифротексті, в 50% випадків починатиметься з парної позиції повідомлення, а в інших 50% – з непарної. Тому розподіл частоти біграм також вирівнюється. Зауважимо, що чим більша кількість алфавітів використовується, тим більш рівномірним стає розподіл частот.

Криптографічне перетворення $\tilde{X}$, яке визначається ключем $\tilde{k} = (X_1, X_2, \dots X_n)$, де$X_i \in S(Z_m)$ та існує хоча б одна пара $i \neq j$, для якої $X_i \neq X_j$, називається **шифром колоною заміни або шифром багатоалфавітної підстановки** періоду n, якщо система рівнянь шифрування послідовних n-грам вихідного тексту $(t_1, t_2, \dots, t_n) \to (s_1, s_2, \dots, s_n)$ має вигляд: $s_i = X(t_i)$, де $i = 1, 2, \dots, n$ [1].

Вагоме місце в класичній криптографії посідає шифр Беласо, де використовуються 26 різних алфавітів, які можна подати у вигляді таблиці, кожний рядок якої містить один з алфавітів, отриманих шляхом зсуву стандартного алфавіту на певну кількість позицій (табл. 1.5).

Таблиця 1.5 – Алфавіти методу Беласо

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
…																									
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Над буквами вхідного повідомлення, яке розміщується по горизонталі, записується ключ. Для шифрування букви, яка знаходиться у верхньому рядку таблиці алфавітів, за допомогою відповідної букви ключа визначається рядок таблиці. Далі відбувається заміна букви відкритого тексту на букву шифротексту, яка міститься прямо над нею у визначеному рядку. Наприклад, відкрита буква N шифрується буквою Q за допомогою букви ключа D (табл. 1.6).

Таблиця 1.6 – Приклад шифрування методом Беласо

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
…																									
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Шифр Беласо вважався незламним, до тих пір, поки Ф. Казіскі не опублікував книгу, в якій описав метод визначення довжини ключа багатоалфавітного шифру.

Суть полягала в тому, щоб знайти повторювані послідовності букв у шифротексті. Звичайно, деякі з них можуть виникнути випадково, особливо біграми, але поява більшості пов'язана з тим, що одні і ті ж букви відкритого тексту зашифровані однією і тією ж частиною ключа. Чим довша повторювана послідовність, тим менш ймовірно, що це випадковість. Якщо дві повторювані послідовності букв зашифровані однією і тією ж частиною ключа, то відстань між ними має бути кратною довжині ключа. Відстань вимірюється від першого символа одного входження до першого символа іншого входження.

Отже, основними кроками в методі Казіскі є [4]:

1. Знаходження в шифротексті послідовності букв, які повторюються;
2. Обчислення найбільшого спільного дільника цих значень, який і буде ключем;
3. Розшифрування за допомогою частотного аналізу кожного набору символів, який зашифрований одним і тим же ключем.

Часто в криптоаналізі використовують індекс збігу, або індекс Фрідмана – це статистичний метод для оцінювання ймовірності того, що дві букви, вибрані навмання з тексту, будуть однаковими.

Суть даного методу полягає в тому, що спочатку текст аналізується та рахується частота кожної букви. Потім обчислюється індекс збігу

$$\overrightarrow{I(x)} = \sum_{i=1}^{m} \frac{f_i(f_i - 1)}{m(m-1)}, \qquad (1.2)$$

де f_i – частота i-тої букви,

m – загальна кількість букв у тексті.

Чим ближче індекс збігу до значення 0.065 (для англійської мови), тим більш ймовірно, що текст зашифровано шифром заміни.

Розглянемо найбільш важливі властивості для оцінки стійкості шифру простої заміни [1].

1. Властивість збереження структури: оскільки вихідний текст шифрується посимвольно, то для шифрування n-грами $(t_1, t_2, \dots, a, \dots, a, \dots t_n)$, у

якої символи на місцях i, j рівні, тобто збігаються, буде отримана n-грама $(s_1, s_2, \dots, b, \dots, b, \dots s_n)$, у якої символи на місцях i, j також рівні.

2. Нехай $\hat{X}$– підстановна матриця, відповідна підстановці X:

$$\hat{X} = \|a_{i,j}\|, \text{ де } a_{i,j} = 1, \text{ якщо } j = X(i), \text{ інакше, } a_{i,j} = 0.$$

Неважко переконатися, що розподіли ймовірностей знаків відкритого $\bar{p} = (p_1, p_2, \dots, p_n)$ і шифрованого $\bar{q} = (q_1, q_2, \dots, q_n)$ текстів пов'язані співвідношенням $\bar{q} = \hat{X}\bar{p}$. Останній вираз можна узагальнити на випадок n різних підстановок $(X_1, X_2, \dots X_n)$, рівноймовірно і незалежно, які обирають для зашифрування відкритого тексту: $\bar{q} = n^{-1}\left(\sum_i \hat{X}_i\right)\bar{p}$.

З наведеного виразу випливає, що зі збільшенням різноманітності підстановок, які застосовуються в процесі шифрування, статистичні властивості шифротексту наближаються до ідеального випадкового розподілу. Швидкість цієї конвергенції залежить від внутрішньої структури підстановок.

Одним з найпростіших видів багатоалфавітних підстановок є шифр Цезаря – це своєрідний «зсув» алфавіту на певну кількість позицій [2]

$$C_m = \left\{C^{(k)} \in S(Z_m):\ C^{(k)} = \begin{pmatrix} i \\ i + k \bmod m \end{pmatrix}\right\},\ k = 0,1, \dots, m-1$$

Хоча такі шифри легко реалізувати, вони дуже вразливі до розшифрування. Навіть невеликий фрагмент зашифрованого тексту дозволяє легко підібрати ключ і відновити вихідне повідомлення, тобто сімейство шифрів Цезаря характеризується досить низькою стійкістю до криптоаналізу.

Серед шифрів заміни виділяються більш складні системи, які замість окремих символів оперують групами символів (біграмами, триграмами тощо). Прикладами таких шифрів є шифр Плейфера та шифр Хілла. Хоча вони забезпечують вищий рівень криптостійкості, їхня реалізація є більш складною і вимагає більшої довжини ключа.

1.2 Шифри перестановки. Ґрати Кардано, магічні квадрати

Другою важливою категорією методів шифрування з використанням секретного ключа є перестановочні шифри, що базуються на зміні порядку елементів повідомлення [1], які можуть містити слова, склади, букви, окремі цифри або біти, що кодують букви.

Перестановка S визначає новий порядок елементів множини $\{1, \dots, n\}$. У криптографії перестановки використовують для зміни порядку символів у відкритому тексті, створюючи у такий спосіб шифротекст.

Для того, щоб показати, що символ переміщений з позиції $\sigma(i)$ в позицію i, де $1 \le i \le n$, використовується запис $\Sigma = \big(\sigma(1), \dots, \sigma(n)\big)$. Водночас число різних перестановок цілих чисел $\{1, \dots, n\}$ дорівнює $n! = 1 \cdot$

$2 \ldots (n-1)n$. На практиці дану величину для достатньо великих n зручно оцінювати за допомогою формули Стірлінга [1]

$$n! = \sqrt{2\pi n} \cdot n^n e^{-n}\big(1 + o(1)\big). \quad (1.3)$$

Нехай $T = \{t_1, t_2, \ldots, t_n\}$ – множина відкритого тексту, що складається з n елементів. Тоді шифрування за допомогою перестановки задається як

$$t_i \to s_i = t_{\sigma(i)}, 1 \le i \le n.$$

Для шифрування за допомогою перестановки Σ текст довжини N ділиться на блоки по n символів. До кожного блока застосовується перестановка, а останній блок, якщо його довжина менша за n, доповнюється деякими символами алфавіту.

Потрібно зазначити, що для кожного блока відкритого тексту може бути застосована індивідуальна перестановка Σ_k. Це приводить до значного збільшення кількості можливих варіантів ключів (до $N(n!)$, де N – кількість блоків, що шифруються, а n – розмір блока). Такий підхід до шифрування, що передбачає використання різних перестановок для різних блоків, застосовується в апаратурі для захисту мовних повідомлень.

На прикладі структурної схеми, наведеної на рисунку 1.1, можна говорити, що генератори потокових шифрів А і Б апаратури засекречування мозаїчного типу створюють послідовність перестановок Σ_k та Σ_k^{-1}, за допомогою яких здійснюється перетворення відкритого і зашифрованого повідомлення відповідно [1].

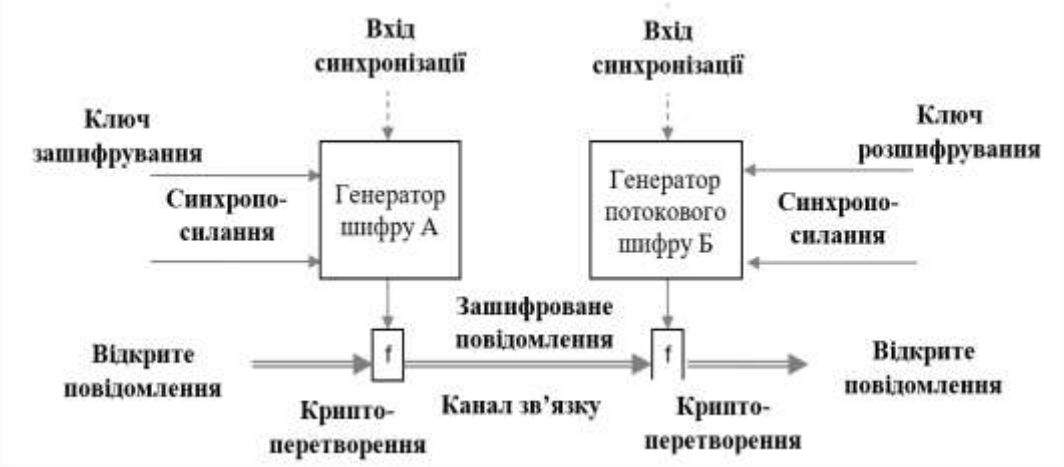


Рисунок 1.1 – Схема лінійного засекречування

Запропонована на рисунку 1.1 схема [1] містить такі основні компоненти:

– генератори потокових шифрів типу А і Б зі своїми відповідними криптографічними модулями;

– систему комунікацій, що забезпечує передачу даних по каналу зв’язку.

Шифр перестановки, за своєю суттю, порушує стійкі сполучення символів у відкритому тексті. При цьому кількість однакових символів у шифрованому тексті залишається такою ж, як і в початковому тексті. Це означає, що ймовірності появи окремих символів у відкритому і шифрованому текстах збігаються, тобто,

$$\forall i,\ j \text{ має місце } p_j = P\{t_i = j\} = P\{s_i = j\} = q_j.$$

Слабкою стороною шифру перестановки є його вразливість до статистичного криптоаналізу. Достатня довжина шифротексту дозволяє виявити статистичні особливості, характерні для мови відкритого тексту, що дозволяє визначити параметри перестановки. Отже, без додаткових заходів шифр перестановки не забезпечує достатній рівень криптостійкості, адже навіть велика кількість можливих ключів не гарантує надійний захист даних.

Підстановка X степеня m на множині A з m елементів – це таке відображення, яке ставить кожному елементу множини A у відповідність один елемент цієї ж множини. Це відображення можна подати у вигляді таблиці, де в першому рядку записані елементи множини A, а в другому – їхні образи при підстановці.

Одним з найпростіших і найстаріших перестановочних шифрів є маршрутна перестановка, яка не використовує ключ. Секретність полягає в тому, що повідомлення записується в прямокутник з використанням одного маршруту, а зчитується з використанням іншого.

Маршрути можна будувати довільним чином, рухаючись з будь-якого кута прямокутника по одному з напрямків: горизонталі, вертикалі, діагоналі, спіралі. Один з можливих прикладів відображено в табл. 1.7.

Таблиця 1.7 – Приклад маршрутної перестановки

21	26	30	33	35	4	3	2	1
22	27	31	34	8	7	6	5	42
23	28	32	12	11	10	9	39	43
24	29	16	15	14	13	37	40	44
25	20	19	18	17	36	38	41	45

Повідомлення вписуємо в жовті комірки в порядку зростання, а зчитуємо по стовпцях.

Розглянемо інший метод, який має назву одиничної перестановки за ключем.

Суть методу полягає в тому, що ключем слугує слово, букви якого нумеруються в алфавітному порядку і розташовуються в порядку зростання для одержання шифрованої фрази.

Наприклад, якщо за ключ обрати слово «ВЕРШИНА», то одержимо шифрувальну таблицю 1.8.

Таблиця 1.8 – Приклад одиночної перестановки за ключем

В	Е	Р	Ш	И	Н	А
2	3	6	7	4	5	1
З	М	Е	Н	І	Н	О
О	О	Ч	Я	Й	А	Р
Р	Я	І	Я	Д	Д	О
Е	В	Р	З	И	Г	Ю

А	В	Е	И	Н	Р	Ш
1	2	3	4	5	6	7
О	З	М	І	Н	Е	Н
Р	О	О	Й	А	Ч	Я
О	Р	Я	Д	Д	І	Я
Ю	Е	В	И	Г	Р	З

Отже, відкритий текст «ЗОРЕ_МОЯ_ВЕЧІРНЯЯ _ЗІЙДИ_НАД_ГОРОЮ» було вписано у стовпці початкової таблиці, а в результаті перестановки стовпців отримали шифротекст «ОЗМІНЕНРО-ОЙАЧЯОРЯДДІЯЮЕВИГРЗ».

У випадку повторень однієї і тієї ж букви в ключі, входження нумеруються зліва направо.

Для підвищення прихованості шифру одиночної перестановки можна використати метод подвійної перестановки. Для цього в таблицю вписується відкритий текст і переставляються спочатку стовпці, а потім рядки. При розшифруванні виконується зворотний порядок перестановок.

Приклад подвійної перестановки повідомлення «ЗУСТРІНЕМОСЬ_ЗАВТРА_О_СЬОМІЙ», ключем якої слугують номери стовпців 25314 та номери рядків 32415, відображається в таблиці 1.9.

Таблиця 1.9 – Приклад подвійної перестановки

	2	5	3	1	4
3	З	У	С	Т	Р
2	І	Н	Е	М	О
4	С	Ь	З	А	В
1	Т	Р	А	О	С
5	Ь	О	М	І	Й

	1	2	3	4	5
3	Т	З	С	Р	У
2	М	І	Е	О	Н
4	А	С	З	В	Ь
1	О	Т	А	С	Р
5	І	Ь	М	Й	О

	1	2	3	4	5
1	О	Т	А	С	Р
2	М	І	Е	О	Н
3	Т	З	С	Р	У
4	А	С	З	В	Ь
5	І	Ь	М	Й	О

Як результат отримаємо зашифроване повідомлення «ОТАСРМІЕОНТЗСРУАСЗВЬЬМЙО».

Існує перестановка з використанням генератора випадкових чисел. За такої умови потрібно для кожної букви повідомлення згенерувати по одному випадковому числу. Далі, зліва направо, обираються усі букви з номером один, потім – з номером два і далі, до тих пір поки не вичерпаємо усі букви.

Говорячи про випадкові числа, варто згадати селекторну перестановку. Ідея якої полягає в тому, що потрібно розбити повідомлення на приблизно рівні частини, а потім об’єднати їх у випадковій послідовності. Розглянемо приклад. Нехай відкритий текст складається з 50 букв. Поділимо його на три частини. Припустимо, що маємо генератор випадкових чисел, який з рівною ймовірністю генерує цифри 0, 1 та 2, і що як ключ використовується початкове значення. Для визначення розміру кожної частини повідомлення згенеруємо перші 50 випадкових цифр і підрахуємо кількість вхо-

джень кожної з них. Припустимо, що згенеровано 15 нулів, 18 одиниць і 17 двійок. Отже, повідомлення поділяється на три частини: P_0 довжиною 15, P_1 довжиною 18 і P_2 довжиною 17. Під час шифрування кожного разу, коли генератор видає 0, беремо наступну букву з P_0, якщо 1 – букву з P_1, а коли 2 – букву з P_2. Так відбувається процес шифрування відкритого тексту при використанні методу селекторної перестановки.

Проаналізуємо клас шифрів перестановки, які називаються решітками або ж Ґратами Кардано, які мають вигляд квадратних таблиць, де чверть осередків прорізана таким чином, що при чотирьох поступових поворотах за годинниковою стрілкою на певний кут, вони покривають весь квадрат. Наприклад, маємо вхідне повідомлення: СПРИЯТЛИВІ_УМОВИ. Спочатку розмістимо його у квадрат 4×4

С	П	Р	И
Я	Т	Л	И
В	І	_	У
М	О	В	И

Позначимо зірочками непрорізані осередки, а повороти здійснюватимемо за годинниковою стрілкою на зазначений нижче кут.

*	*	С	*
*	*	*	П
*	Р	*	*
И	*	*	*

Я	*	*	*
*	Т	*	*
*	*	*	Л
*	*	И	*

*	*	*	В
*	*	І	*
_	*	*	*
*	У	*	*

*	М	*	*
О	*	*	*
*	*	В	*
*	*	*	И

Я	М	С	В
О	Т	І	П
_	Р	В	Л
И	У	И	И

0' 90' 180' 270'

Отже, зашифроване повідомлення матиме вигляд: ЯМСВОТІП_РВЛИУИИ

Цікавим класом шифрів є так звані магічні квадрати, заповнені послідовними натуральними числами, починаючи з одиниці, так, що сума чисел в кожному рядку, стовпці та діагоналі – однакова. Причому, зі збільшенням розміру квадрата, стрімко зростає кількість магічних квадратів.

Нехай потрібно зашифрувати повідомлення: ТЕХНІЧНИЙ_ЗАХИСТ за допомогою магічного квадрата розміром 4×4

2	11	7	14
13	8	12	1
16	5	9	4
3	10	6	15

Тоді першу букву вхідного повідомлення (Т) занесемо в комірку квадрата з числом 1, другу букву (Е) запишемо в комірку з номером 2 тощо, в

результаті отримаємо шифрувальний квадрат

Е	З	Н	И
Х	И	А	Т
Т	І	Й	Н
Х	_	Ч	С

У такий спосіб зашифроване повідомлення матиме вигляд ЕЗНИХИ-АТТІЙНХ_ЧС.

Загальним методом розшифрування перестановочних шифрів є метод множинних анаграм. Для його застосування потрібно перехопити декілька повідомлень однакової довжини (як мінімум три). Якщо для перестановки символів у цих повідомленнях використовувався один і той самий ключ, то перші букви всіх повідомлень будуть знаходитися на однакових позиціях у всіх шифротекстах, так само як і другі, і всі наступні букви.

Цей факт можна використати так: спочатку на одну паперову смужку виписати всі перші букви шифротекстів, на другу смужку – всі другі букви і далі. Кількість смужок має відповідати довжині шифротекстів. Чим більше повідомлень, тим довшими будуть смужки і тим вища ймовірність успіху розшифрування.

Для проведення статистичного аналізу шифру подвійної перестановки за допомогою таблиці імовірності біграм, потрібно [3]:

1. Укласти шифротекст у квадрат розміром $n \times n$;
2. Розрахувати логарифми ймовірностей для всіх можливих сполучень стовпців. При цьому потрібно використовувати таблицю логарифмів з основою два ймовірностей біграм англійського тексту;
3. Для визначення правильного розташування стовпців здійснити оцінювання імовірності пар сполучень різних стовпців і обрати сполучення з найбільшою імовірністю.

Розглянемо приклад використання описаного криптоаналізу шифрів перестановки.

Нехай маємо зашифроване повідомлення: QRUEGBRISAFMCEYN.

Розмістимо даний шифротекст у квадрат розміром 4×4:

	1	2	3	4
1	Q	R	U	E
2	G	B	R	I
3	S	A	F	M
4	C	E	Y	N

Оцінимо статистично, які стовпці будуть знаходитися поруч. Для цього розрахуємо логарифми ймовірностей для всіх можливих поєднань першого стовпця з іншими. Ймовірність того, що два стовпці будуть розташовані поруч, дорівнює добутку логарифмів біграм у відповідних рядках

цих стовпців. Оскільки в таблиці наведені логарифми біграм (дод. А) [5], їх достатньо додати і обрати комбінації стовпців з найбільшим логарифмом ймовірності (це зумовлено тим, що логарифм – функція, що постійно зростає).

У такий спосіб обчислимо імовірності всіх можливих пар стовпців:

p(1-2) = p(QR) + p(GB) + p(SA) + p(CE) = 0 + 5,36 + 12,1+ 12,7 = 30,16
p(1-3) = p(QU) + p(GR) + p(SF) + p(CY) = 10,9 + 11,3+ 8,62 + 9,93 = 40,75
p(1-4) = p(QE) + p(GI) + p(SM) + p(CN) = 0,69 + 11+ 10,1+ 4,9 = 26,69
p(2-1) = p(RQ) + p(BG) + p(AS) + p(EC) = 6,18 + 3,56 + 13 + 12,4 = 35,14
p(2-3) = p(RU) + p(BR) + p(AF) + p(EY) = 11+ 10,6 + 10,6 + 11,4 = 43,6
p(2-4) = p(RE) + p(BI) + p(AM) + p(EN) = 13,7 + 10,9 + 11,9 + 13,4 = 49,9
p(3-1) = p(UQ) + p(RG) + p(FS) + p(YC) = 5,73 + 11+ 8,16 + 8,48 = 33,37
p(3-2) = p(UR) + p(RB) + p(FA) + p(YE) = 12,3+ 9,63+ 11,1+ 11,4 = 44,43
p(3-4) = p(UE) + p(RI) + p(FM) + p(YN) = 11,2 + 12,7+ 6,27 + 9,11= 39,28
p(4-1) = p(EQ) + p(IG) + p(MS) + p(NC) = 9,52 + 11,8+ 10,6 + 12,2 = 44,12
p(4-2) = p(ER) + p(IB) + p(MA) + p(NE) = 13,9 + 10,4 + 12,3 + 12,8 = 38,4
p(4-3) = p(EU) + p(IR) + p(MF) + p(NY) = 9,23 + 11,9 + 7,59 + 11,1 = 39,82

Далі оцінимо імовірність пар сполучень різних стовпців і виберемо найбільше значення.

p(1-2-3-4) = 30,16 + 39,28 = 69,44 43,6
p(1-2-4-3) = 30,16 + 39,82 = 69,98 49,9
p(1-3-2-4) = 40,75 + 49,9 = 90,65 44,43
p(1-3-4-2) = 40,75 + 38,4 = 79,15 39,28
p(1-4-2-3) = 26,69 + 43,6 = 70,29 38,4
p(1-4-3-2) = 26,69 + 44,43 = 71,12 39,82
p(2-1-3-4) = 35,14 + 39,28 = 74,42 40,75
p(2-1-4-3) = 35,14 + 39,82 = 74,96 26,69
p(2-3-1-4) = 43,6 + 26,69 = 70,29 33,37
p(2-3-4-1) = 43,6 + 44,12 = 69,98 39,28
p(2-4-1-3) = 49,9 + 40,75 = 90,65 44,12
p(2-4-3-1) = 49,9 + 33,37 = 83,27 39,82
p(3-1-2-4) = 33,37 + 49,9 = 83,27 30,16
p(3-1-4-2) = 33,37 + 38,4 = 71,77 26,69
p(3-2-1-4) = 44,43 + 26,69 = 71,12 35,14
p(3-2-4-1) = 44,43 + 44,12 = 88.55 49,9
p(3-4-1-2) = 39,28 + 30,16 = 69,44 44,12
p(3-4-2-1) = 39,28 + 35,14 = 74,42 38,4
p(4-1-2-3) = 44,12 + 43,6 = 69,98 30,16
p(4-1-3-2) = 44,12 + 44,43 = 88.55 40,75
p(4-2-1-3) = 38,4 + 40,75 = 79,15 35,14
p(4-2-3-1) = 38,4 + 33,37 = 71,77 43,6
p(4-3-1-2) = 39,82 + 30,16 = 69,98 33,37
p(4-3-2-1) = 39,82 + 35,14 = 74,96 44,43

З множини отриманих значень виберемо найбільше p(1-3-2-4) = 90,65 та розмістимо стовпці в такому порядку: 1-3-2-4:

	1	3	2	4
1	Q	U	R	E
2	G	R	B	I
3	S	F	A	M
4	C	Y	E	N

Отримана комбінація не може допомогти розшифрувати повідомлення, тому далі аналізуємо найбільше значення імовірності пар сполучень різних стовпців, а саме: p(2-4-1-3) = 90,65. Розмістивши стовпці в порядку 2-4-1-3, отримаємо таку таблицю:

	2	4	1	3
1	R	E	Q	U
2	B	I	G	R
3	A	M	S	F
4	E	N	C	Y

Підібравши потрібне розташування рядків (2-3-1-4) матимемо зв'язний текст: BIGRAMS FREQUENCY.

Отже, в даному підрозділі було проаналізовано різні типи шифрів перестановки та розглянуто можливі методи їх криптоаналізу.

1.3 Гамування та криптоаналіз шифру Віженера

Окремим випадком шифру однозначної заміни є **шифр гамування**, коли за базовий набір підстановок використовується сімейство підстановок Цезаря, а вихідний текст вигляду $T = t_1, t_2, \ldots$ перетвориться в зашифрований текст $S = s_1, s_2, \ldots$ за правилом [1]

$$s_i = C^{(\gamma_i)}(t_i) = (t_i + \gamma_i)\, mod\, m. \qquad (1.4)$$

Залежно від властивостей гами, шифри гамування поділяються на періодичні та випадкові. Періодичною називають гаму для якої існує деяке $p > 0$ таке, що $\forall i\ \gamma_i = \gamma_{i+p}$. Періодичні гами мають повторювану структуру, що впливає на криптостійкість шифру. Потужність ключового простору такого шифру дорівнює m^p. В свою чергу, псевдовипадкові послідовності з великим періодом є ефективним способом уникнення цієї проблеми і забезпечення високої стійкості шифру.

Одним із поширених методів генерації гамових послідовностей є використання псевдовипадкових числових генераторів [1].

Якщо довжина ключа шифру дорівнює довжині повідомлення, а ключ генерується випадково та використовується лише один раз, то такий

метод шифрування називається шифром Вернама або системою одноразового використання [6].

Оскільки в цій системі шифрування ключі використовувалися лише один раз та записувалися на фізичні носії (спочатку на перфострічки, потім у спеціальні блокноти), то її часто називають «системою одноразових блокнотів». Такий блокнот містив відривні сторінки, на кожній з яких була надрукована таблиця з випадковими цифрами або буквами. Причому для кожного символа повідомлення кожен символ з блокнота використовується тільки один раз. У випадку повного використання таблиці, вона відривається і знищується.

На початку шифрування методом гамування знаки відкритого тексту і гами записуються в два рядки, утворюючи вертикальні біграми знаків.

Розрізняють два види гамування – модульне і табличне.

При модульному гамуванні додаються відповідні числові значення букв відкритого тексту та гами за модулем m, де модуль дорівнює кількості знаків в алфавіті.

У разі табличного гамування вертикальні пари, утворені комбінацією букв відкритого тексту і гами, замінюються на знаки шифротексту відповідно до деякої таблиці, яка обов'язково має бути так званим «латинським квадратом». Це означає, що довільний її рядок і стовпець мають являти собою перестановку знаків заданого алфавіту (або чисел від 1 до m), крім того, елементи, розміщені в кожнім стовпці і рядку даної таблиці, мають бути різними.

Для отримання шифрованого тексту (S) можна використовувати один з трьох способів накладання гами (Г) на відкритий текст (О):

– додавання гами і тексту ($S = Г + О$);

– віднімання з тексту гами ($S = О - Г$);

– віднімання з гами тексту ($S = Г - О$).

Причому, під операціями «додавання» і «віднімання» матимемо на увазі як звичайні операції за модулем m, так і застосування замість них відповідних таблиць.

Процедура розшифрування, ґрунтується на обернених перетвореннях: $О = S - Г$, $О = S + Г$, $О = Г - S$, або, відповідно, на обернених таблицях.

Зашифруємо відкритий текст: НАВЧАННЯ_ЦЕ_СВІТЛО, використовуючи як ключ слово ЗАХИСТ. У цьому випадку шифрування зручніше виконувати шляхом додавання за модулем 33 цілих чисел, що є номерами букв в алфавіті, починаючи з одиниці (табл. 1.10) [1].

Таблиця 1.10 – Приклад шифрування гамуванням

Н	А	В	Ч	А	Н	Н	Я	Ц	Е	С	В	І	Т	Л	О
18	1	3	28	1	18	18	33	27	7	22	3	12	23	16	19
З	А	Х	И	С	Т	З	А	Х	И	С	Т	З	А	Х	И
10	1	26	11	22	23	10	1	26	11	22	23	10	1	26	11
28	**2**	**29**	**6**	**23**	**8**	**28**	**1**	**20**	**18**	**11**	**26**	**22**	**24**	**9**	**30**
Ч	**Б**	**Ш**	**Д**	**Т**	**Є**	**Ч**	**А**	**П**	**Н**	**И**	**Х**	**С**	**У**	**Ж**	**Щ**

У результаті отримаємо такий шифротекст: ЧБШДТЄЧАПНИХ-СУЖЩ.

Шифр гамування забезпечує вирівнювання ймовірностей зустрічі знаків шифрованого тексту. Очевидно, якщо маємо розподіл ймовірностей зустрічання знаків гами $(g_1, \ldots, g_m)$, то розподіл імовіростей символів шифрованого тексту описується такою системою рівнянь [1]:

$$q_1 = p_1 g_m + p_2 g_{m-1} + \cdots + p_m g_1,$$
$$q_2 = p_1 g_1 + p_2 g_m + \cdots + p_m g_2,$$
$$\ldots$$
$$q_m = p_1 g_{m-1} + p_2 g_{m-2} + \cdots + p_m g_{m1}.$$

Шифр на основі одноразової гами забезпечує ідеальну стійкість. Однак його використання для захисту конфіденційної інформації часто є практично неможливим через труднощі з генерацією та розповсюдженням великих обсягів ключових даних.

Крім шифрів модульного гамування існують шифри, що використовують інші операції для відображення пари (t, γ), відкритий текст-гама в знак шифрованого тексту: $c = f_v(t, \gamma)$. Функція f_v являє собою обернене табличне перетворення, що використовується на такті шифрування v, такі шифри називаються шифрами табличного гамування [1].

Розглянемо шифр модульного гамування з рівнянням, де гама є періодичною послідовністю символів алфавіту, яка отримана шляхом періодичного повторення певного ключового слова. Проаналізуємо задачу криптоаналізу цього шифру, який має назву шифр Віженера. Дана задача зводиться до проведення двох етапів [7]:

1. Визначення на основі тесту Казіскі μ – довжини ключового слова;
2. Визначення ключового слова.

Нагадаємо, що тест Казіскі ґрунтується на простому припущенні: якщо два однакові «куски» відкритого тексту знаходяться на відстані, кратній μ, то вони будуть зашифровані однаково. Тому в шифротексті шукають не менше трьох повторень і визначають відстань між ними [8].

Нехай d_1, d_2, ... – знайдені відстані між повтореннями і d – найбільший спільний дільник цих чисел. Чим більше повторень має текст, тим більш ймовірно, що μ збігатиметься з d.

Для точнішого визначення μ можна використати індекс збігу – імовірність збігу двох довільних символів в рядку.

Розглянемо досить довгий рядок $\vec{x}$ з m символів. Якщо f_i – кількість i-го символа в рядку $\vec{x}$, то індекс збігу обчислюється за формулою

$$\overrightarrow{I(x)} = \sum_{i=1}^{m} \frac{f_i(f_i - 1)}{m(m-1)}. \qquad (1.5)$$

Нехай $\vec{x}$– рядок змістовного відкритого тексту. Припустимо, що в $\vec{x}$ букви з'являються незалежно одна від одної в будь-якому місці тексту з ймовірностями $p_0, \dots p_{n-1}$, де p_i – ймовірність появи букви i в змістовному тексті. Тоді ймовірність того, що дві випадково вибрані букви з x збігаються з $i \in Z_n$, обчислюється за формулою

$$I_c(x) \approx \sum_{i=0}^{n-1} p_i^2. \tag{1.6}$$

Для текстів x англійською мовою $I_c(\text{x}) \approx 0{,}066$. Аналогічні наближення можна отримати й для інших мов (табл. 1.11) [8].

Таблиця 1.11 – Індекси збігів європейських мов

Мова	Англійська	Французька	Німецька	Італійська	Іспанська
$I_c(x) \approx$	0,0662	0,0778	0,0762	0,0738	0,0775

Якщо ж $\vec{x}$ – випадковий рядок, то ймовірність появи кожного символа дорівнює

$$p_i = \frac{1}{m}. \tag{1.7}$$

Тоді для оцінення індексу збігів поліалфавітного шифру використовують формулу

$$\overrightarrow{I(x)} = \sum_{i=1}^{m} \frac{1}{m^2}. \tag{1.8}$$

Оскільки значення індексу збігів для відкритого тексту та для поліалфавітного шифру значно відрізняються, то це дозволяє, на основі індексу збігів, визначити, чи є отриманий текст результатом простої перестановки відкритого тексту, чи він зашифрований поліалфавітним шифром.

Нехай маємо шифротекст $y = y_1 y_2 \dots y_n$. Випишемо даний шифротекст з періодом μ.

$$\begin{array}{cccc} Y_1^{\downarrow} & Y_2^{\downarrow} & \dots & Y_\mu^{\downarrow} \\ y_1 & y_2 & \dots & y_\mu \\ y_{\mu+1} & y_{\mu+2} & \dots & y_{2\mu} \\ y_{2\mu+1} & y_{2\mu+2} & \dots & y_{3\mu} \\ \dots & \dots & \dots & \dots \end{array}$$

Позначимо стовпці отриманої таблиці через $Y_1^{\downarrow}, \dots, Y_\mu^{\downarrow}$.

Розглянемо два варіанти: якщо μ – це дійсна довжина ключового слова, або ж якщо μ не збігається з довжиною ключового слова.

В першому варіанті кожний стовпець таблиці $Y_i^{\downarrow}$, є ділянкою відкритого тексту, зашифрованого простою заміною, в такому разі $I_c(Y_i^{\downarrow}) \approx 0{,}066$, (для англійської мови) при будь-якому i.

З іншого боку, розглядаючи другий варіант, стовпці $Y_i^{\downarrow}$ будуть більш «випадковими», оскільки вони є результатом шифрування фрагментів відкритого тексту деяким багатоалфавітним шифром. Тоді $I_c\ (Y_i^{\downarrow})$ буде ближче до числа $1/26 \approx 0{,}038$ (для англійської мови).

В більшості випадків різниця значень $I_c(x)$ для змістовних відкритих текстів і випадкових послідовностей букв [1] (для англійської мови – 0,066 і 0,038) дозволяє встановити точне значення μ.

Далі переходимо до знаходження ключового слова, використовуючи взаємний індекс збігів ($[\![MI]\!]\_c\ (x, y)$) – ймовірність того, що випадково вибрана буква з x збігається з випадково вибраною буквою з y, де $x = (x_1, \dots, x_n)$, $y = (y_1, \dots, y_m)$ – два рядки букв деякого алфавіту, з довжинами m та $\acute{m}$, відповідно.

Нехай $f_0 f_1 \dots f_n$ і ${f_0}^1 {f_1}^1 \dots {f_{n-1}}^1$ – кількість входження букв алфавіту в першому і другому рядках, відповідно, тоді взаємний індекс збігів буде обчислюватися за формулою

$$MI_c(x, y) = \sum_{i=0}^{n-1} \frac{f_i \cdot f_i^1}{m \cdot m'}. \tag{1.9}$$

Нехай $k = (k_1, \dots k_\mu)$ – ключове слово. Оцінимо взаємні індекси збігів $MI_c\left(Y_i^{\downarrow}, Y_j^{\downarrow}\right)$.

Пригадаємо, що $Y_3^{\downarrow}$ – результат зашифрованого фрагмента відкритого тексту простої заміни, що визначається підстановкою для деякого $s \in 0,\ n-1$ (числа беруться за модулем n)

$$\begin{pmatrix} 0 & 1 & 2 & \dots & n-s & \dots & n \\ s & s+1 & s+2 & \dots & 0 & \dots & s-1 \end{pmatrix}$$

Ймовірність того, що $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$ довільна пара букв дорівнює 0, має вигляд $p_{h-s_i} * p_{h-s_j}$ (де p_a – ймовірність появи букви a у відкритому тексті). Ймовірність того, що обидві букви є 1, дорівнює $p_{h-si+1} * p_{h-sj+1}$ далі аналогічно. На основі цих суджень отримаємо

$$MI_c\left(Y_i^{\downarrow}, Y_j^{\downarrow}\right) \approx \sum_{h=0}^{n-1} p_{h-s_i} * p_{h-s_j} = \sum_{h=0}^{n-1} p_h * p_{h+(s_i-s_j)}. \tag{1.10}$$

Відмітимо, що сума в правій частині останньої рівності залежить тільки від різниці $(s_i - s_j)\, mod\, n$, яку назвемо відносним зсувом $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$, причому

$$\sum_{j=0}^{n-1} p_j \cdot p_{(j+s)} mod\ n = \sum_{j=0}^{n-1} p_j \cdot p_{(j-s)}\ mod\ n. \qquad (1.11)$$

Тому $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$ з відносними зсувами s і n-s мають однакові взаємні індекси збігів. У таблиці 1.12 відображено значення сум для англійської мови.

Таблиця 1.12 – Взаємний індекс збігів при зсуві s

Зсув s	0	1	2	3	4	5	6
$MI_c(x,y) \approx$	0,066	0,039	0,032	0,034	0,044	0,033	0,036
Зсув s	7	8	9	10	11	12	13
$MI_c(x,y) \approx$	0,039	0,034	0,034	0,038	0,045	0,039	0,043

Звернемо увагу на те, що ненульові зсуви дають взаємні індекси збігів, які змінюються в межах від 0,032 до 0,045, в той час як при нульовому зсуві індекс $MI_c(x,y) \approx 0{,}066$. Дане спостереження дозволяє визначити величини відносних зсувів $s_i - s_j$ стовпців $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$. Для цього відмітимо, що при деякому значенні $s(i,j) \in 0, n-1$ стовпець $Y_j^{s(i,j)\downarrow}$, отриманий з $Y_j^{\downarrow}$ додаванням до кожного його елемента числа $S(i,j)$ (за модулем n), має нульовий відносний зсув з $Y_j^{\downarrow}$.

Нехай $Y_j^{0\downarrow}, Y_j^{1\downarrow}, \dots, Y_j^{n-1\downarrow}$ – результати шифрування $Y_j^{\downarrow}$ кожною з простих замін, тоді визначимо взаємні індекси збігів

$$MI_c\left(Y_i^{\downarrow}, Y_j^{s\downarrow}\right) = \sum_{h=0}^{n-1} \frac{f_h \cdot f_{h-s}^1}{m \cdot m'}. \qquad (1.12)$$

Якщо $s = s_i - s_j$ (відносного зсуву $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$), то взаємний індекс збігів має бути (для англійської мови) близький до 0,066, оскільки відносний зсув $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$ дорівнює нулю. Якщо ж $s \neq s_i - s_j$, то взаємний індекс збігів має лежати в межах 0,032–0,045.

Використовуючи описаний метод, можна зв'язати системою рівнянь відносні зсуви різних пар стовпців $Y_i^{\downarrow}$ і $Y_j^{\downarrow}$. В результаті залишиться 26 (для англійської мови) варіантів для ключового слова, з яких можна вибрати кращий варіант (якщо ключове слово є змістовним).

1.4 Практичні аспекти розділу 1

1.4.1 Шифр Цезаря: принцип роботи, приклади застосування

Однією з давніх класичних систем шифрування є шифр простої заміни Цезаря [9]. Цей шифр заснований на принципі зсуву літер певного алфавіту на заздалегідь узгоджену кількість позицій k та заміні кожного символа відкритого повідомлення на відповідний символ. Так буде отримано новий алфавіт шифротексту, який складається з символів основного алфа-

віту. Значення k є ключем шифру, який буде використовуватися як для процесів шифрування, так і розшифрування, $k \in \{1 \dots N\}$.

Шифрування здійснюється за формулою 1.13

$$E(m) = (m + k) \bmod N, \tag{1.13}$$

де m та k – номери літер, відповідно, відкритого повідомлення та шифротексту;

k – ключ шифру;

N – потужність алфавіту.

Для розшифрування застосовується обернений алгоритм за формулою 1.14

$$D(c) = (c - k) \bmod N. \tag{1.14}$$

Для кращого уявлення про криптоалгоритми продемонструємо сучасну навчальну програму CrypTool 2 [10], яка дозволяє використовувати та аналізувати криптоалгоритми.

Зашифруємо нижченаведене Вхідне повідомлення:

Cryptography is the process of hiding or coding information so that only the person a message was intended for can read it. The art of cryptography has been used to code messages for thousands of years and continues to be used in bank cards, computer passwords, and ecommerce. Modern cryptography techniques include algorithms and ciphers that enable the encryption and decryption of information, such as 128-bit and 256-bit encryption keys. Modern ciphers, such as the Advanced Encryption Standard (AES), are considered virtually unbreakable [11].

Ключ для шифрування оберемо k = 7. Тобто, зсув літер вхідного алфавіту буде на 7 позицій праворуч (табл. 1.13).

Таблиця 1.13 – Позиції літер вхідного та вихідного алфавітів

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	0	1	2	3	4	5	6

На рисунку 1.2 продемонстровано процес шифрування з k = 7.

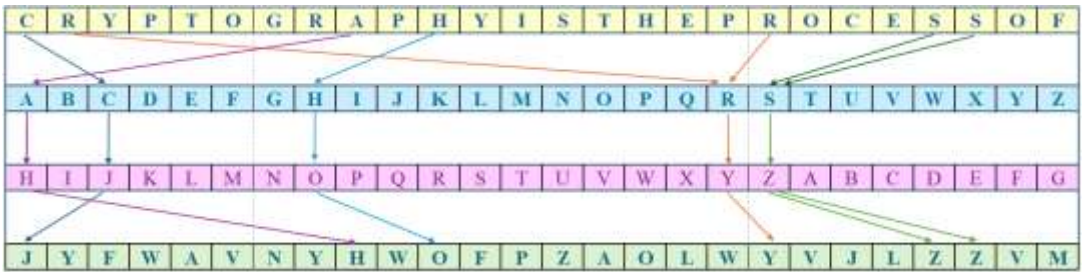


Рисунок 1.2 – Приклад використання шифру простої заміни

Відповідні поля заповнюємо вхідними даними та обираємо процес шифрування (рис. 1.3).

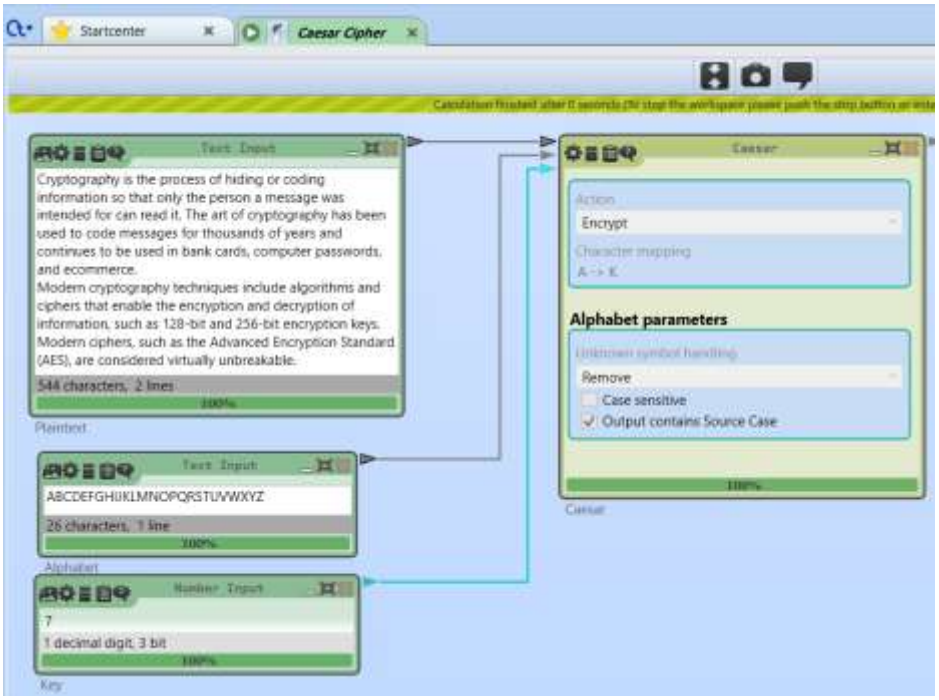


Рисунок 1.3 – Процес шифрування

Після чого запускаємо програму та отримаємо зашифроване повідомлення шифром Цезаря (рис. 1.4).

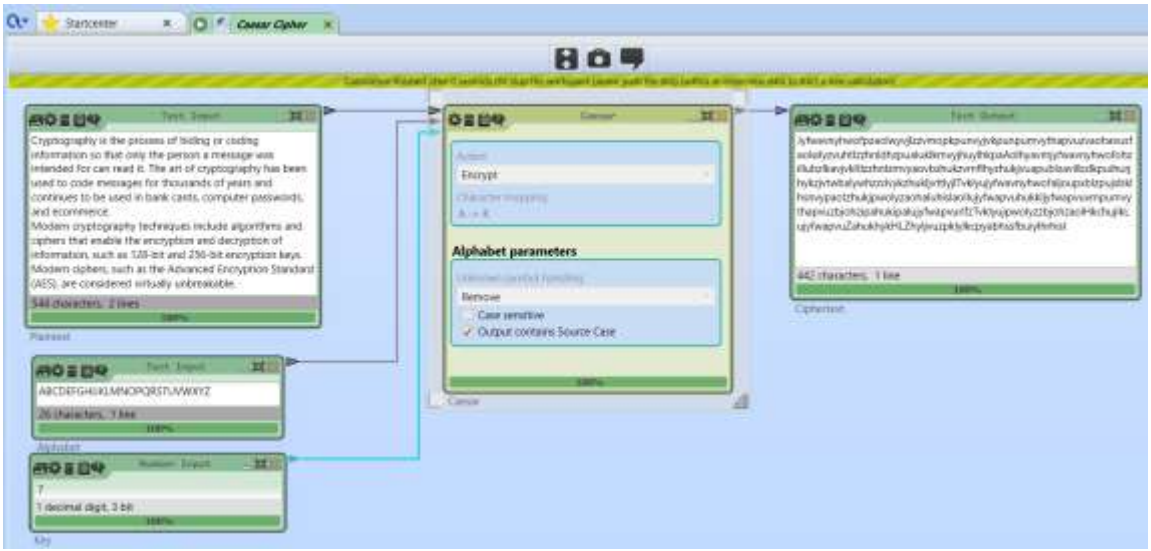


Рисунок 1.4 – Результат шифрування

Таким чином, було отримано Зашифроване повідомлення.

JyfwavnyhwofpzaolwyvjlzzvmopkpunvyjvkpunpumvythapvuzvaohavusfaolwlyzvuhtlzzhnldhzpualuklkmvyjhuylhkpaAolhyavmjyfwavnyhwofohzillubzlkavjvkltlzzhnlzmvyaovbzhukzvmflhyzhukjvuapublzavilbzlkpuihurjhykzjvtwbalywhzzdvykzhukljvttlyjlTvklyujyfwavnyhwofaljoupxblzpujsbklhsnvypaotzhukjpwolyzaohaluhislaollujyfwapvuhukkljyfwapvuvmpumvythapvuzbjohzipahukipalujyfwapvurlfzTvklyujpwolyzzbjohzaolHkchujlkLujyfwapvuZahukhykHLZhyljvuzpklylkcpyabhssfbuiylhrhisl

Недоліком шифрів заміни є вразливість до частотного аналізу, результатом проведення якого буде отримання секретного ключа.

Шифри заміни вразливі до атак із відкритим текстом, мають обмежену розміром алфавіту кількість ключів та вразливі до атаки грубої сили й атаки звичайного перебору.

Кількість можливих ключів дорівнює розміру заданого алфавіту. Використання літер A–Z як алфавіт дозволяє отримати 26 різних ключів, причому 26-й ключ втрачає сенс, оскільки він буде ставити у відповідність кожну літеру самій собі. Маючи лише 25 значущих ключів, буде досить легко перебрати всі можливі варіанти ключів, поки не буде знайдено правильний (аналіз грубою силою).

Шифри заміни також можна легко зламати за допомогою частотного аналізу.

У криптоаналізі частотний аналіз – це дослідження частоти літер або груп літер у зашифрованому тексті. Метод використовується як допоміжний засіб для розкриття класичних шифрів [2].

Із цієї дефініції можна припустити, що цей метод може бути використаний для розкриття шифру Цезаря.

Можна швидко визначити величину зсуву, вивчивши зміщення певних елементів після побудови графіка частот літер у зашифрованому тексті і знаючи передбачуваний розподіл цих літер в оригінальній мові відкритого тексту.

Отже, для визначення величини зсуву методом частотного аналізу потрібно мати таблицю і графік ймовірностей літер англійського алфавіту відкритого тексту (табл. 1.14, рис. 1.5) [12] та графік частотного аналізу зашифрованого тексту (рис. 1.6).

Таблиця 1.14 – Відносні частоти та інформаційна вага літер латинського алфавіту

N (j)	Символ (j)	P (j)	Log_2 P (j)	N (j)	Символ (j)	P (j)	Log_2 P (j)
0	**A**	0,0816	3,62	13	**N**	0,0675	3,89
1	**B**	0,0149	6,07	14	**O**	0,0751	3,74
2	**C**	0,0278	5,17	15	**P**	0,0193	5,70
3	**D**	0,0425	4,56	16	**Q**	0,0009	10,12
4	**E**	0,1270	2,98	17	**R**	0,0598	4,06
5	**F**	0,0228	5,45	18	**S**	0,0632	3,98
6	**G**	0,0201	5,64	19	**T**	0,0906	3,46
7	**H**	0,0609	4,04	20	**U**	0,0276	5,18
8	**I**	0,0696	3,84	21	**V**	0,0098	6,67
9	**J**	0,0015	9,38	22	**W**	0,0236	5,41
10	**K**	0,0077	7,02	23	**X**	0,0015	9,38
11	**L**	0,0402	4,64	24	**Y**	0,0197	5,67
12	**M**	0,0241	5,37	25	**Z**	0,0007	10,48

На рисунку 1.5 зображено графік розподілу літер латинського алфавіту.

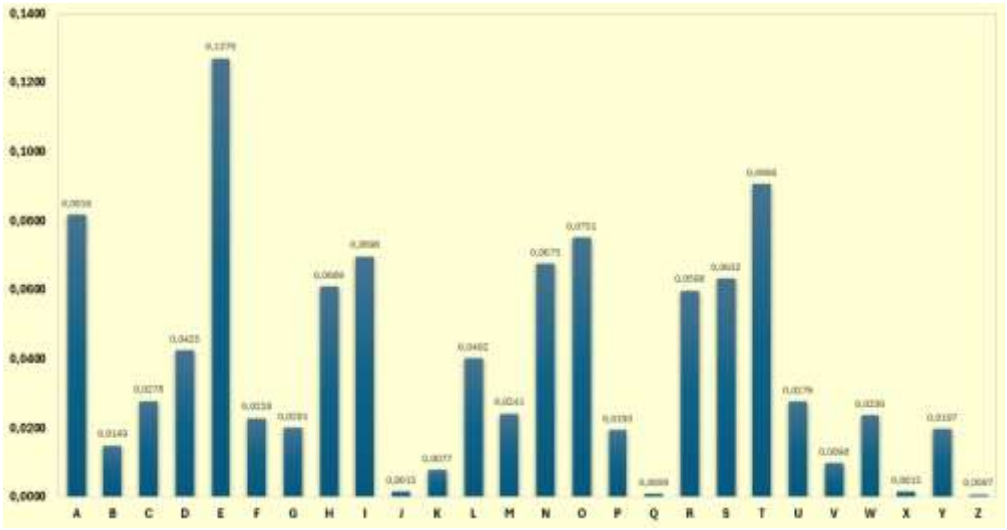


Рисунок 1.5 – Розподіл літер в англійській мові відкритого тексту

Для проведення частотного аналізу зашифрованого тексту запустимо модуль «Frequency analysis» в програмі CrypTool 2 та в «Text Input» введемо зашифрований текст (рис. 1.6).

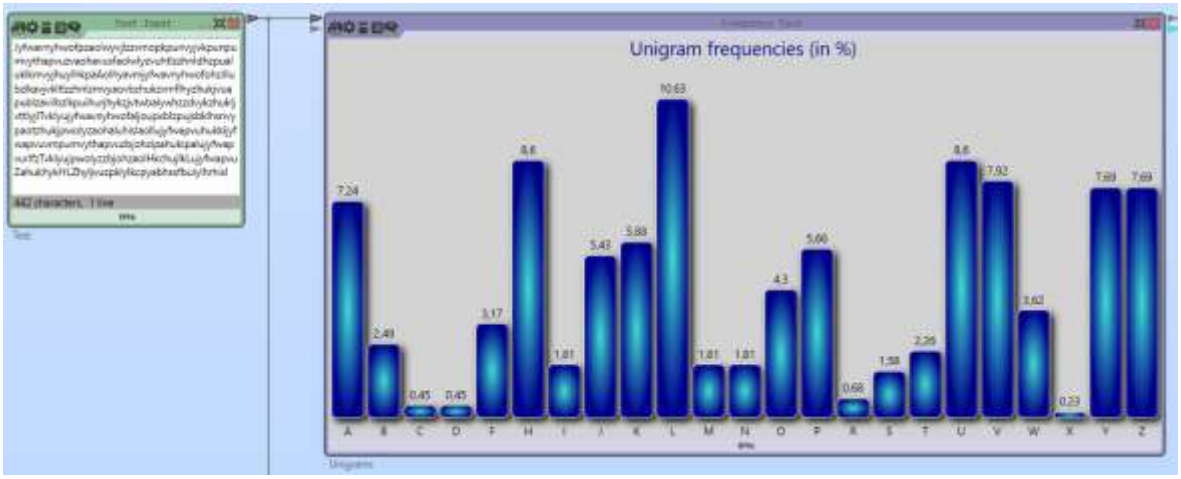


Рисунок 1.6 – Розподіл літер у зашифрованому тексті

За результатами аналізу графіків частотного розподілу у відкритому тексті та зашифрованому, визначено, що у відкритому тексті з найбільшою частотою є літери «E», «T», «A», а в зашифрованому тексті – літери «L», «U», «H».

Ці літери мають такі позиції:

«E» = 4, «T» = 19, «A» = 0;

«L» = 11, «U» = 20, «H» = 7.

За формулою 1.2 визначимо значення зсуву, тобто ключ шифру

$$11 - 4 = 7$$

$$7 - 0 = 7$$

Спробуємо розшифрувати повідомлення із отриманим значення $k = 7$ (рис. 1.7).

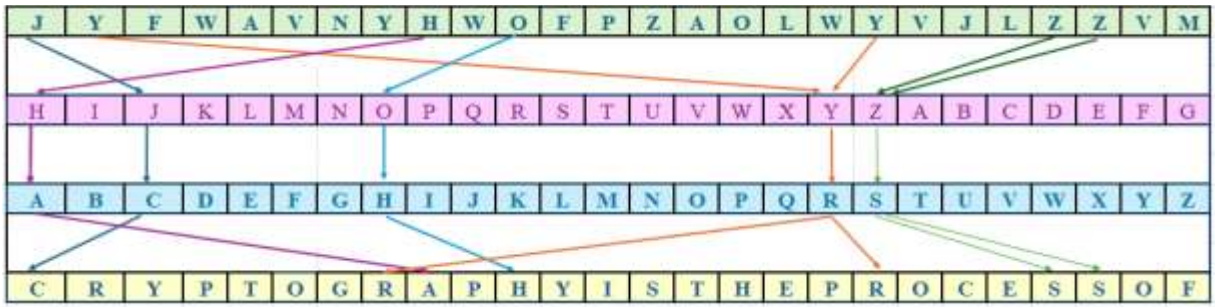


Рисунок 1.7 – Приклад розшифрування повідомлення

Розшифруємо зашифроване повідомлення в CrypTool 2 методом частотного аналізу (рис. 1.8).

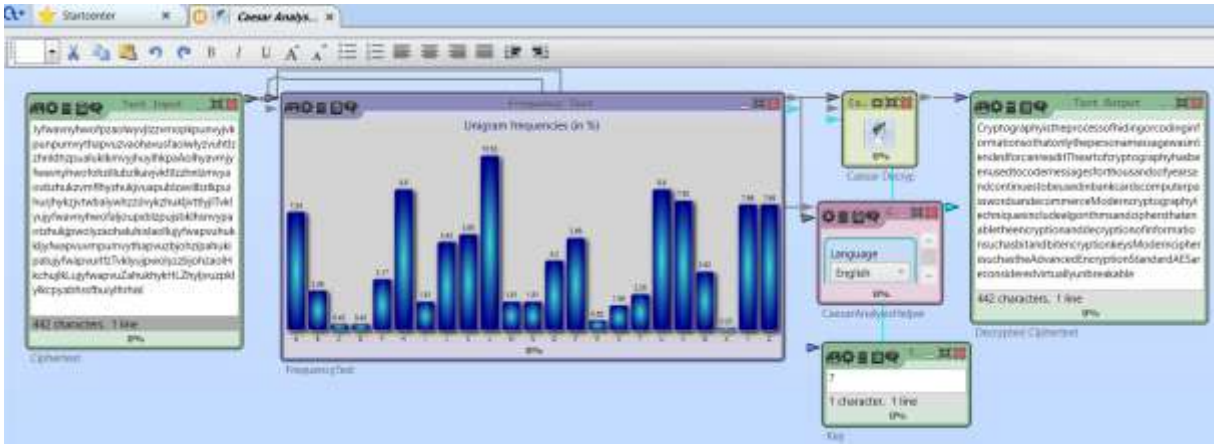


Рисунок 1.8 – Розшифрування повідомлення методом частотного аналізу

Для результативного проведення частотного аналізу шифрів заміни буде недостатньо отримати доволі короткий шифротекст, через те, що розподіл ймовірностей буде, майже, однаковий.

1.4.2 Шифрування методом Віженера: практичне застосування та криптоаналіз

Шифр Віженера був розроблений у 16-му столітті французьким криптологом Блезом де Віженером. Він заснований на використанні шифру Цезаря та використовує просту форму поліалфавітної заміни [9].

Поліалфавітний шифр використовує декілька алфавітів для шифрування повідомлень. На відміну від моноалфавітних шифрів, де кожна літера відкритого тексту замінюється однією й тією самою літерою шифрованого тексту, у поліалфавітних шифрах одна і та ж літера відкритого тексту може замінюватися різними літерами залежно від її позиції у повідомленні.

Шифрування вихідного тексту здійснюється за допомогою квадрата Віженера (табл. 1.15) [13].

Таблиця 1.15 – Таблиця Віженера

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
0	**A**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	**B**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	**C**	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	**D**	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	**E**	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	**F**	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	**G**	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	**H**	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	**I**	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	**J**	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	**K**	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	**L**	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	**M**	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	**N**	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	**O**	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	**P**	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	**Q**	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	**R**	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	**S**	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	**T**	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	**U**	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	**V**	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	**W**	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	**X**	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	**Y**	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	**Z**	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Таблиця складається з 26 алфавітів. Кожен алфавіт циклічно зсувається на 1 символ вліво порівняно з попереднім алфавітом, що відповідає 26 можливим варіантам шифру Цезаря. Величина зсуву визначається ключем, набором літер, яким відповідають числа, відповідні позиції літери в алфавіті. Наприклад, якщо маємо ключ CHW, то літери відкритого тексту зсуваються на 2, 7 і 22 позиції. Послідовність 2, 7, 22 повторюється, доки не буде зашифровано весь відкритий текст. Наприклад, слово BASIC на ключі CHW буде зашифроване як DHOKJ (рис. 1.9).

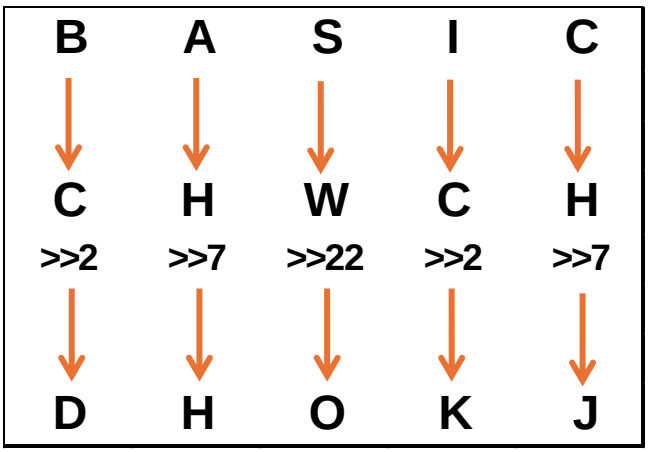


Рисунок 1.9 – Шифрування методом Віженера

Шифрування та розшифрування здійснюється, відповідно, за формулами 1.15 та 1.16:

$$E_i = (P_i + K_i) \bmod N, \tag{1.15}$$

$$D_i = (E_i - K_i + N) \bmod N, \tag{1.16}$$

де i – номер позиції літер алфавіту $[0, 1, \ldots . N]$;

P_i та E_i – літери, відповідно, відкритого повідомлення та шифротексту;

K_i – ключова послідовність;

N – потужність алфавіту.

Зашифруємо нижченаведене Вхідне повідомлення.

THEVIGENERECIPHERISAMETHODOFENCRYPTINGALPHABETI
CTEXTBYUSINGASERIESOFINTERWOVENCAESARCIPHERSBASEDO
NTHELETTERSOFAKEYWORDITEMPLOYSAFORMOFPOLYALPHABET
ICSUBSTITUTIONFIRSTDESCRIBEDBYGIOVANBATTISTABELLASOIN
FIFTEENFIFTYTHREETHECIPHERISEASYTOUNDERSTANDANDIMPLE
MENTBUTITRESISTEDALLATTEMPTSTOBREAKITUNTILEIGHTEENSI
XTYTHREETHREECENTURIESLATERTHISEARNEDITTHEDESCRIPTIO
NLECHIFFREINDECHIFFRABLEMANYPEOPLEHAVETRIEDTOIMPLEM
ENTENCRYPTIONSCHEMESTHATAREESSENTIALLYVIGENERECIPHE
RSINEIGHTTEENSIXTYTHREEFRIEDRICHKASISKIWASTHEFIRSTTOP
UBLISHAGENERALMETHODOFDECIPHERINGVIGENERECIPHERSINT
HENINETEENTHCENTURYTHESCHEMEWASMISATTRIBUTEDTOBLAI
SEDEVIGENEREANDSOACQUIREDITSPRESENTNAME

Для зручного шифрування повідомлення вручну виділимо лише рядки, що є літерами обраного ключа, в цьому прикладі ключ «CRYPTOGRAPHY» (табл. 1.16).

Таблиця 1.16 – Таблиця шифрування методом Віженера

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
2	**C**	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
17	**R**	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
24	**Y**	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
15	**P**	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
19	**T**	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
14	**O**	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
6	**G**	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
17	**R**	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
0	**A**	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
15	**P**	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
7	**H**	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
24	**Y**	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X

Щоб зашифрувати літеру T за допомогою літери C, знаходимо літеру шифротексту на перетині рядка C і стовпця T, це буде літера V.

Для розшифрування знаходимо рядок з літерою ключа, наприклад, літера H і в цьому ж рядку знаходимо літеру шифротексту L. Обираємо літеру відкритого тексту даного стовпця літеру E.

Зашифруємо відкрите повідомлення в програмі CrypTool 2 за допомогою модуля «Vigenere Chipher».

Відповідні поля заповнюємо вхідними даними, вводимо ключ «CRYPTOGRAPHY» та обираємо процес шифрування (рис. 1.10).

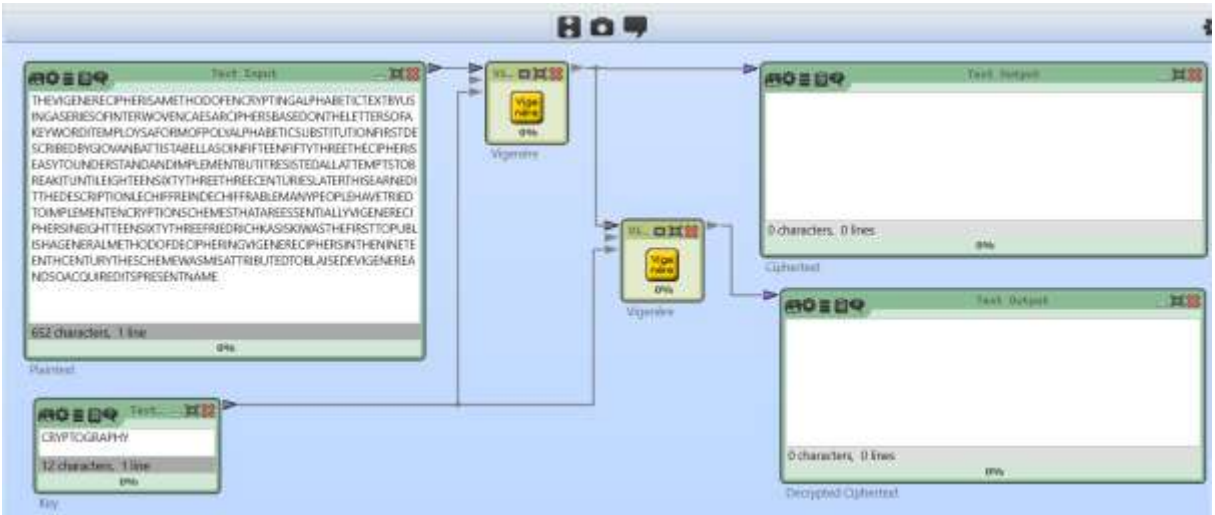


Рисунок 1.10 – Введення вхідних даних

Наступним кроком буде введення параметрів алфавіту (рис. 1.11).

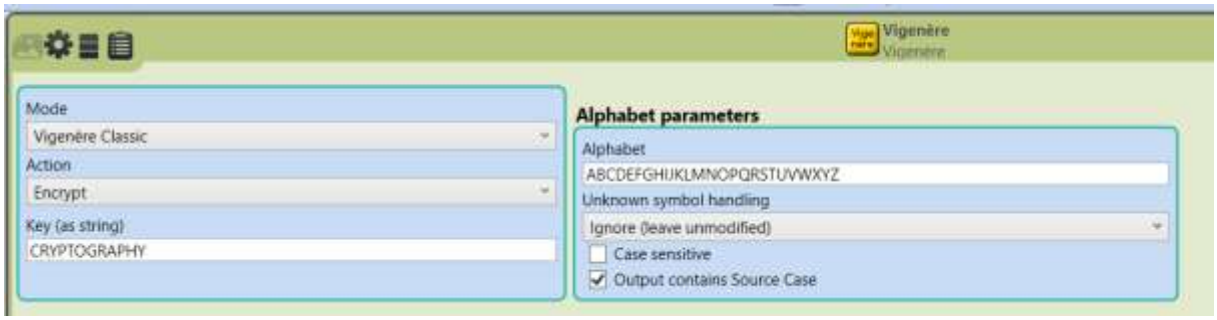


Рисунок 1.11 – Введення ключа та параметрів алфавіту

Після чого запускаємо програму та отримаємо зашифроване повідомлення шифром Віженера (рис. 1.12).

Рисунок 1.12 – Процес шифрування методом Віженера

Отже, отримано Зашифроване повідомлення має вигляд:

VYCKBUKEEGLAKGFTKWYRMTAFQUMUXBIIYEAGPXYAIVGSEIPA
VVVIUMAJICNYUVPXXGUWICACTNMKXBIREHHPEZNWXFYSAHLB
QERWXZKKTTYQQWYZXMCFRSPRGDNAHMYRFDYKQWNDEMGCP
WHZGKGRLIHJTXASVZMCYWXJTSLQEIGQXRHPGXVTCEZPMHOJTP
ICNCYHHWTWIUACGEDXYHEKHGLCVYCRBDNVRXZCCJWIHITUEG
ZRCEBPGRODPALKGERQNHOKRTZGUKCSTZRRTILKRKQIHPXVAZP
RWERXESOXHILCPJGMMMZYRTLRJICTVSTKUGPCUCYIXFZYIHLYT
ECSBHZYESLQEIGEMWUELTJFKWDGXWTUEROGHWPPUZKDACFN
GFNAXVGMEIYGGURDBAVCEBLLVVLRKMVKIDUQEYCBXGZYAIHP
GVQHXBZZAASWXZETGSXVCXWFGIQXGSOXHIACGEQXQHEKHGL
CHIGTWFOTHZHQKJIXPOYKHTMGTJRIHDASLXZFCXCCXFGCMTAF
QUMUWSIZPWLPKEEKBUKEEGLAKGFTKGOETWLLKECIXSTKHRLL
VLPNMVKJCWLKGNYHFWYRTIYGDLRTWHUSLPPQGUCKBUKEEGL
YPUQDTQWLIGLBKKQEKSYVNIUYOV

Перевагою шифру Віженера є приховування інформації про частоту літер в тексті, водночас для однакових літер відкритого тексту можуть використовуватися декілька літер шифрованого тексту, що залежить від розміру ключа і ключових літер, який циклічно повторюється протягом всього відкритого повідомлення. Тобто, одна й та ж літера відкритого тексту не завжди буде мати повну відповідність літері зашифрованого тексту. Таким чином, маскування частоти літер перешкоджатиме методам частотного аналізу [14].

Недоліком шифру Віженера є те, що алгоритм використовує симетричний ключ, тобто, один секретний ключ як для шифрування, так і для розшифрування. Ключова послідовність циклічно повторюється до тих пір, поки не зрівняється з довжиною відкритого тексту.

Циклічне повторення ключа дає можливість значно швидше визначити його довжину за допомогою тесту Казіскі [15]. Якщо довжина буде відома, зловмиснику достатньо виконати частотний аналіз для різних шифрів Цезаря.

Для проведення криптоаналізу шифру Віженера, насамперед, потрібно визначити довжину ключової послідовності. Для цього запустимо програму CrypTool 2 модуль «Kasiski's Test and Autocorrelation». Результат проведеного тесту зображено на рисунку 1.13.

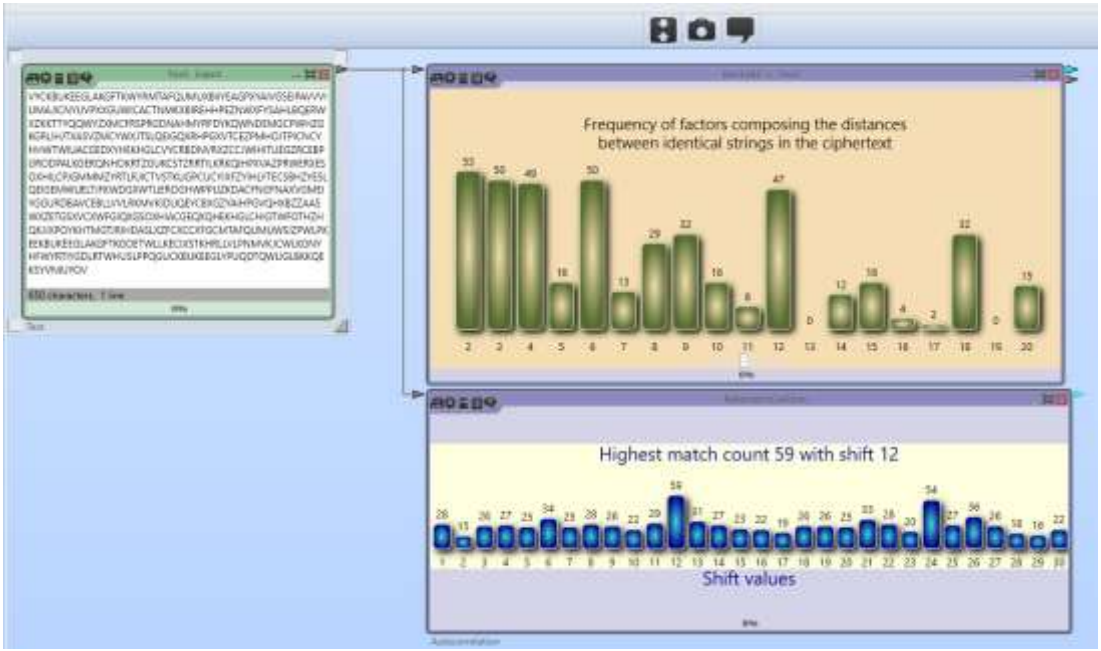


Рисунок 1.13 – Визначення довжини ключової послідовності

Отже, за результатом тесту Казіскі визначено ймовірнісний ключ довжиною 12.

Для проведення аналізу і визначення ключа для зашифрованого тексту запустимо модуль «Vigenere analysis» в програмі CrypTool2 та в «Text Input» введемо зашифрований текст і запустимо програму (рис. 1.14).

Рисунок 1.14 – Введення даних для визначення ключа шифрування

На рисунку 1.15 зображено ключ «CRYPTOGRAPHY», яким було зашифровано відкрите повідомлення.

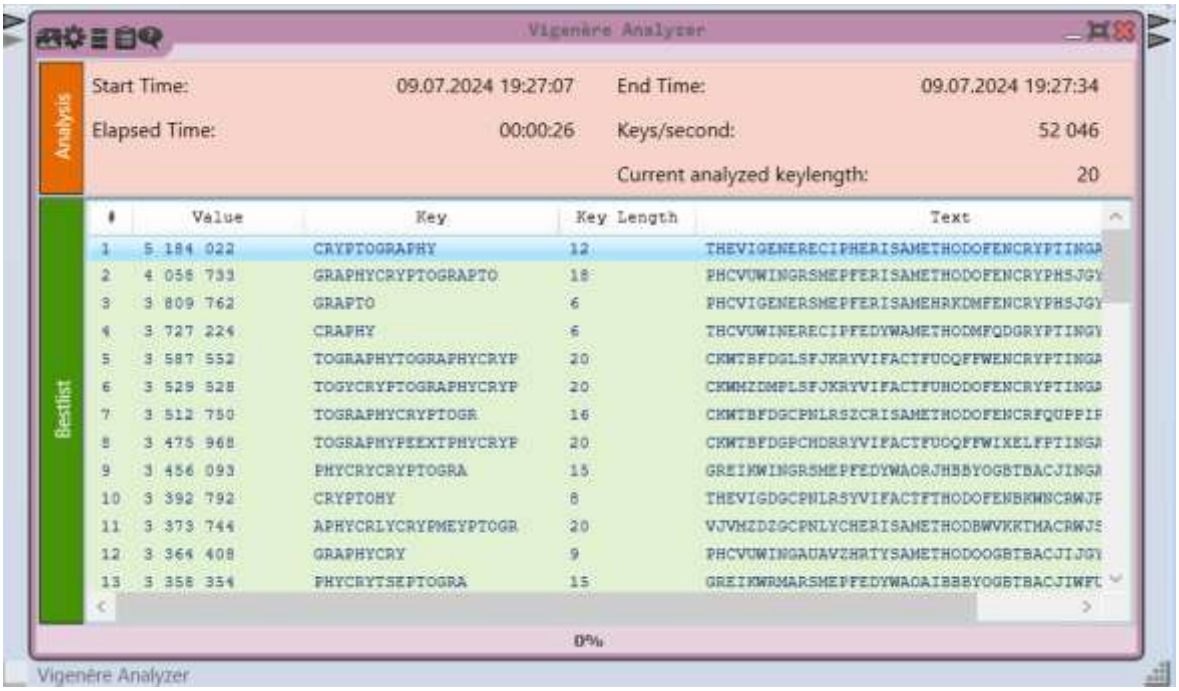


Рисунок 1.15 – Відображення ключа шифрування

Для кращого розуміння криптоаналізу шифру Віженера скористаємось ресурсом [16].

Спочатку зашифруємо вхідне повідомлення (рис. 1.16).

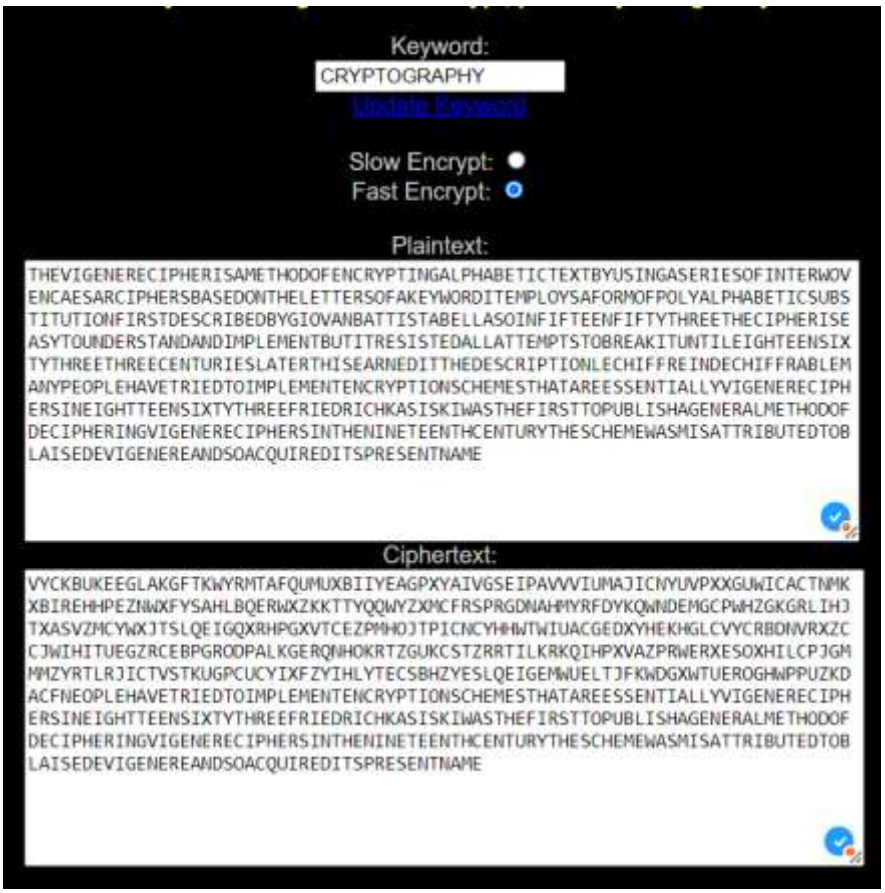
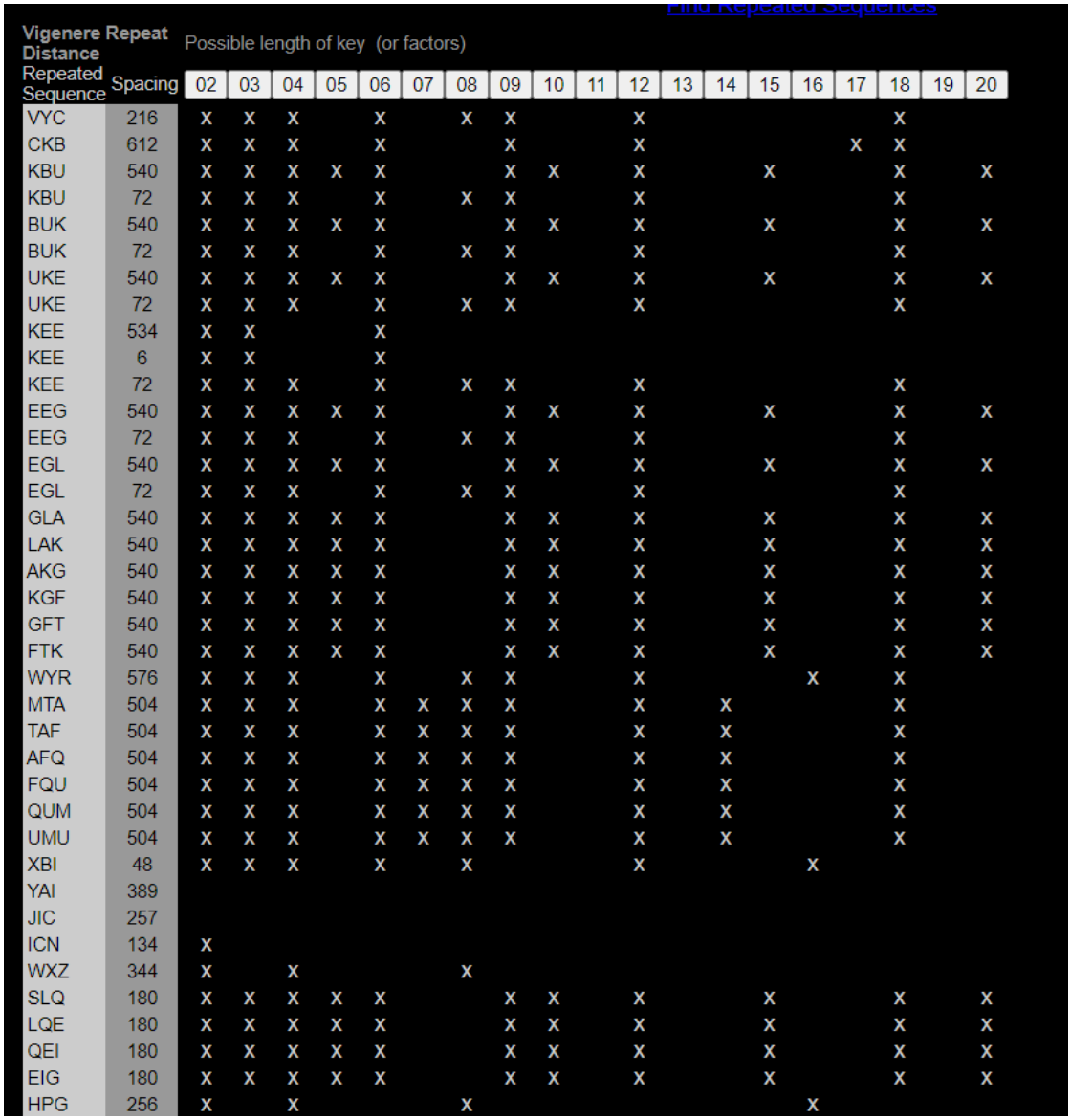


Рисунок 1.16 – Процес шифрування вхідного повідомлення

Для проведення криптоаналізу у відповідне поле вставимо зашифрований текст і запустимо пошук послідовностей літер, які з’являються більше одного разу (рис. 1.17).

Рисунок 1.17 – Виявлення найпоширеніших послідовностей літер

Причиною таких повторів є те, шо деякі послідовності літер у відкритому тексті були зашифровані за допомогою однакових послідовностей ключа. Результат пошуку довжини ключової послідовності зображено на рисунку 1.18.



Vigenere Repeat Distance

Repeated Sequence	Spacing	Possible length of key (or factors) 02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
VYC	216	x	x	x		x		x	x			x						x		
CKB	612	x	x	x		x			x			x					x	x		
KBU	540	x	x	x	x	x			x	x		x			x			x		x
KBU	72	x	x	x		x		x	x			x						x		
BUK	540	x	x	x	x	x			x	x		x			x			x		x
BUK	72	x	x	x		x		x	x			x						x		
UKE	540	x	x	x	x	x			x	x		x			x			x		x
UKE	72	x	x	x		x		x	x			x						x		
KEE	534	x	x			x														
KEE	6	x	x			x														
KEE	72	x	x	x		x		x	x			x						x		
EEG	540	x	x	x	x	x			x	x		x			x			x		x
EEG	72	x	x	x		x		x	x			x						x		
EGL	540	x	x	x	x	x			x	x		x			x			x		x
EGL	72	x	x	x		x		x	x			x						x		
GLA	540	x	x	x	x	x			x	x		x			x			x		x
LAK	540	x	x	x	x	x			x	x		x			x			x		x
AKG	540	x	x	x	x	x			x	x		x			x			x		x
KGF	540	x	x	x	x	x			x	x		x			x			x		x
GFT	540	x	x	x	x	x			x	x		x			x			x		x
FTK	540	x	x	x	x	x			x	x		x			x			x		x
WYR	576	x	x	x		x		x	x			x				x		x		
MTA	504	x	x	x		x	x	x	x			x		x				x		
TAF	504	x	x	x		x	x	x	x			x		x				x		
AFQ	504	x	x	x		x	x	x	x			x		x				x		
FQU	504	x	x	x		x	x	x	x			x		x				x		
QUM	504	x	x	x		x	x	x	x			x		x				x		
UMU	504	x	x	x		x	x	x	x			x		x				x		
XBI	48	x	x	x		x		x				x				x				
YAI	389																			
JIC	257																			
ICN	134	x																		
WXZ	344	x		x				x												
SLQ	180	x	x	x	x	x			x	x		x			x			x		x
LQE	180	x	x	x	x	x			x	x		x			x			x		x
QEI	180	x	x	x	x	x			x	x		x			x			x		x
EIG	180	x	x	x	x	x			x	x		x			x			x		x
HPG	256	x		x				x								x				

Рисунок 1.18 – Результат пошуку довжини ключової послідовності

Для визначення довжини ключа потрібно обрати найпоширеніший коефіцієнт інтервалу, в нашому випадку – 12. Після того, як буде обрано ймовірну довжину ключа 12, отримаємо поки що невизначене ключове слово: L1-L2-L3-L4-L5-L6-L7-L8-L9-L10-L11-L12.

Кожна літера L_i означає відповідний рядок квадрата Віженера для заміни літер відкритого тексту, наприклад, 1-ої, 13-ої, 25-ої, 37-ої, 49-ої тощо.

Таким чином, поліалфавітний шифр складається з 12 одноалфавітних шифрів.

Для того, щоб виконати частотний аналіз, обираємо кожну літеру L_i і запускаємо процес (рис. 1.19).

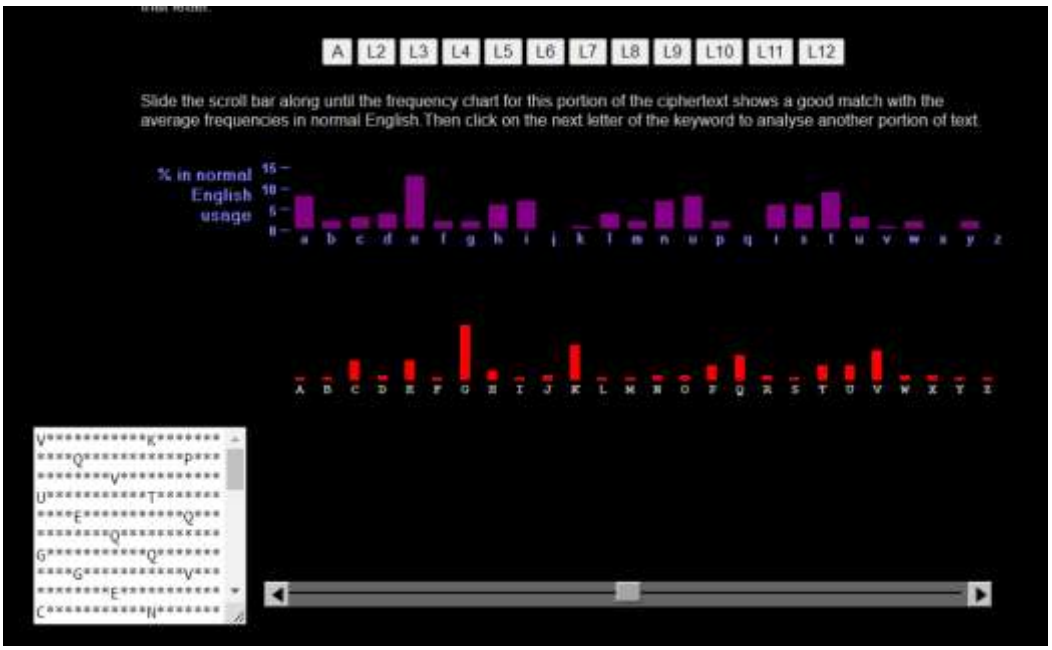


Рисунок 1.19 – Початок проведення частотного аналізу

Пік високої частоти у звичайній англійській мові з літери E перемістився на літеру G в зашифрованому тексті. Змістимо повзунок до повного їх збігу (рис. 1.20).

Рисунок 1.20 – Проведення частотного розподілу

Аналогічні кроки виконаємо щодо решти літер ключової послідовності (дод. Б). В результаті отримаємо шукану ключову послідовність (рис. 1.21).

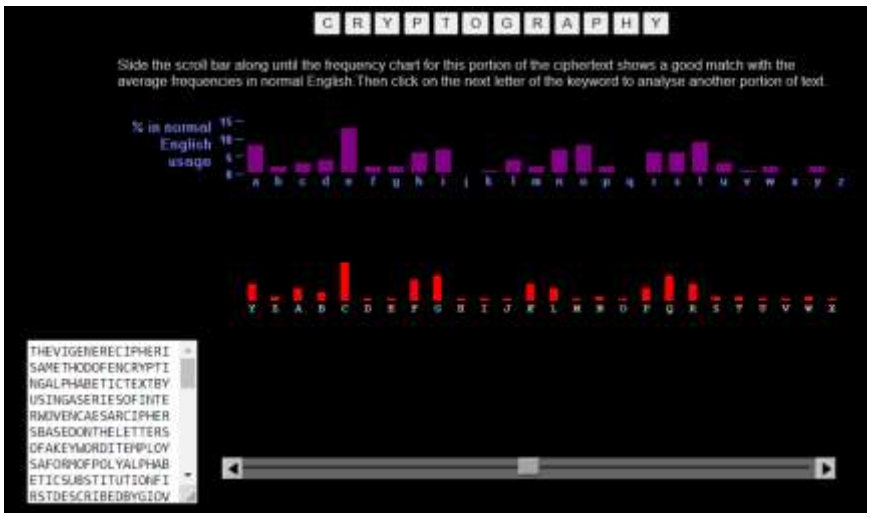


Рисунок 1.21 – Відображення ключової послідовності

Так було визначено ключову послідовність «CRYPTOGRAPHY» за допомогою тесту Казіскі і частотного аналізу кожного з 12 алфавітів, в результаті отримано розшифроване повідомлення, що повністю збігається з відкритим початковим повідомленням (рис. 1.21).

1.5 Контрольні питання

1. Що є предметом вивчення криптографії?
2. Суть математичного об’єкта «підстановка»?
3. Який принцип роботи шифру простої заміни?
4. Як відбувається шифрування за допомогою перестановки?
5. Що таке шифр гамування і як він працює?
6. Опишіть алгоритм шифрування методом Цезаря.
7. Що означає «біграма»?
8. Опишіть метод рандомізації відкритого тексту.
9. Назвіть основні етапи криптоаналізу шифру одноалфавітної заміни.
10. У чому суть методу Беласо?
11. Як частотний аналіз дозволяє відрізнити шифр простої заміни від шифру перестановки?
12. Які основні компоненти схеми лінійного шифрування?
13. Опишіть метод селекторної перестановки.
14. Назвіть основні кроки проведення статистичного аналізу шифру подвійної перестановки.
15. Як зрозуміти термін «період гами» в контексті шифрування?
16. Які різновиди гамування Ви знаєте?
17. У чому суть тесту Казіскі?
18. Яким вимогам має відповідати гама в шифрі Вернама?
19. Що таке індекс збігу і як він обчислюється?
20. Як визначити взаємний індекс збігів?

РОЗДІЛ 2 СИМЕТРИЧНІ КРИПТОСИСТЕМИ

Симетричні криптосистеми є одними з найстаріших та найпоширеніших методів захисту інформації. Вони ґрунтуються на використанні одного ключа для шифрування і розшифрування даних, що робить їх одночасно простими у реалізації та ефективними за швидкодією.

Всю різноманітність існуючих симетричних криптосистем, що використовуються для шифрування вихідних повідомлень, можна звести до класів перетворень, наведених на рис. 2.1 [1].

Рисунок 2.1 – Класифікація симетричних криптосистем

Криптосистеми можна класифікувати на потокові та блокові.

У потокових на кожному такті шифрування застосовується змінний алгоритм, що залежить від початкових ключових даних і порядкового номера поточного такту шифрування.

Блокові шифри працюють з фіксованими блоками даних. Для кожного блока використовується один і той же алгоритм шифрування, який визначається ключем, встановленим на початку процесу.

До найпоширеніших алгоритмів симетричного шифрування належать: DES (Data Encryption Standard), Triple DES (3DES), AES (Advanced Encryption Standard), Blowfish, Twofish, RC4 [2]. Вагоме місце посідають такі національні крипто алгоритми, як: ДСТУ ГОСТ 28147:2009, ДСТУ 7624:2014 («Калина»), «Струмок».

2.1 Американський стандарт шифрування даних DES

Найбільш відомим алгоритмом симетричного шифрування є DES, який був розроблений у 70-х роках XX ст. фахівцями IBM і у 1976 році був прийнятий NIST (National Institute of Standards and Technolody США) як федеральний стандарт Сполучених Штатів [2].

Алгоритм DES відносять до класу блокових шифрів, для яких вихідні дані та результати шифрування подаються у вигляді 64-бітових рядків – блоків. Шифрування і розшифрування в алгоритмі здійснюються за однаковими схемами за винятком процедури формування ключів, що використовуються в циклах його роботи. Ключем шифрування є випадкове число довжиною 56 бітів, що доповнюється для підвищення надійності зберігання та транспортування вісьмома бітами перевірки парності, так що загальна довжина ключа дорівнює 64 бітам. Алгоритм реалізує так звану схему Фейстеля, що передбачає ітераційну обробку двох частин блока вихідних даних. Водночас в кожному циклі роботи алгоритму з використанням підстановок і перестановок реалізуються функції розсіювання (дифузії) і перемішування. Всього в алгоритмі виконується 16 раундів (циклів) (див. рис. 2.1) [1].

Блок вихідних даних після початкової перестановки (IP) (табл. 2.1) розбивається на лівий L_0 і правий R_0 півблоки однакової довжини [17].

Таблиця 2.1 – Перетворення IP

58,	50,	42,	34,	26,	18,	10,	2,
60,	52,	44,	36,	28,	20,	12,	4,
62,	54,	46,	38,	30,	22,	14,	6,
64,	56,	48,	40,	32,	24,	16,	8,
57,	49,	41,	33,	25,	17,	9,	1,
59,	51,	43,	35,	27,	19,	11,	3,
61,	53,	45,	37,	29,	21,	13,	5,
63,	55,	47,	39,	31,	23,	15,	7

Для зашифрування блока відкритого тексту виконуються 16 однотипних циклів, в кожному з яких застосовується свій ключ (K_i), який створюється з основного ключа (K) і називається підключем (рис. 2.2).

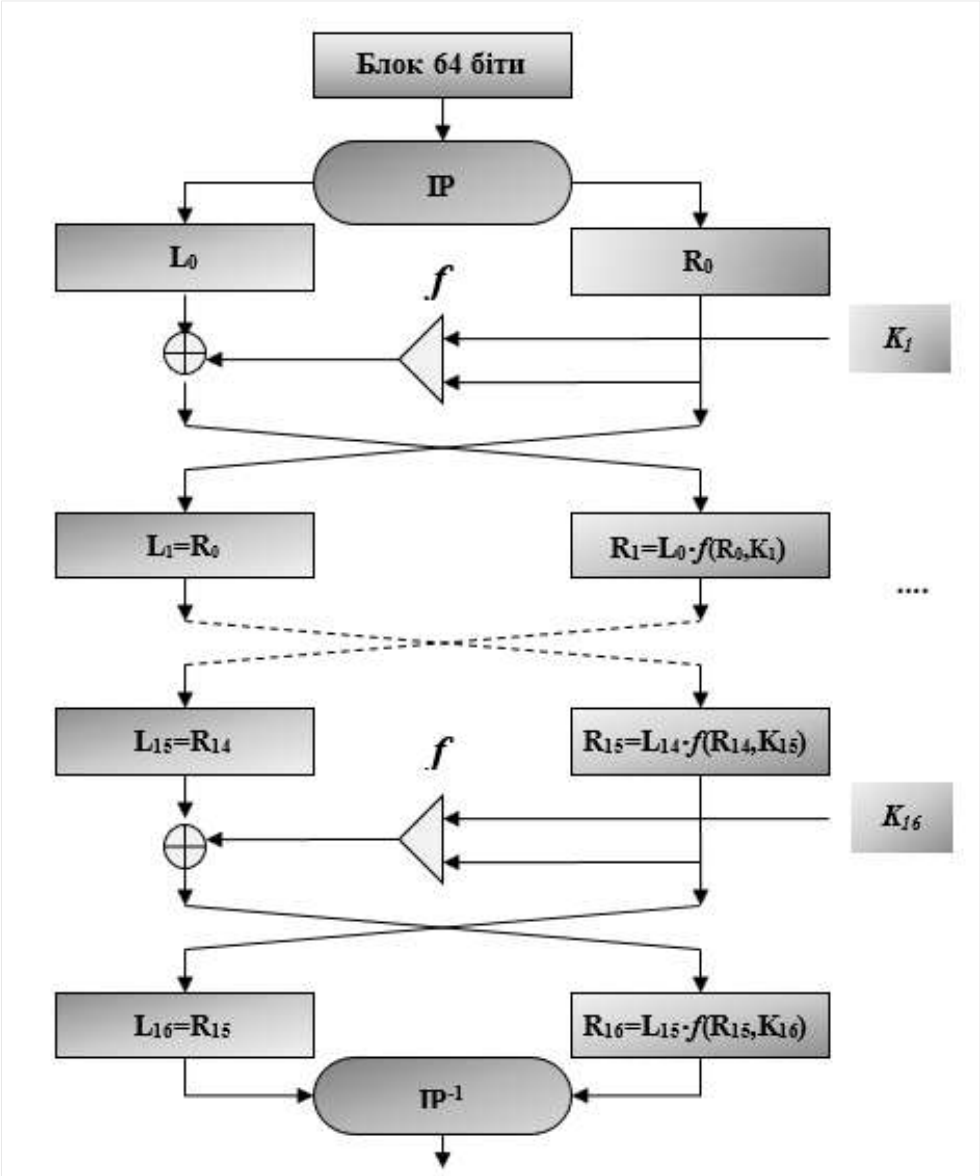


Рисунок 2.2 – Структурна схема алгоритму DES

На кожному з циклів півблоки комбінуються з ключем K_{j+1} відповідного раунду.

Основою шифру DES є несекретна циклова функція

$$Y = f(R, K). \tag{2.1}$$

Якщо (R, L) – вхід у цикл, то перетворення півблоків матиме вигляд

$$R_i = L_{i-1} \oplus f\left(R_{i-1,}\, K_i\right), \text{де } L_i = R_{i-1}\,(i = 1, \dots, 16). \tag{2.2}$$

На початковому етапі 64-бітовий ключ трансформується в 56-бітовий шляхом видалення кожного восьмого біта (табл. 2.2) [17]. Ці біти не впливають на шифрування, але використовуються для контролю парності, що дозволяє перевірити правильність ключа.

Таблиця 2.2 – Перестановка PC-1

57,	49,	41,	33,	25,	17,	9,	1,
58,	50,	42,	34,	26,	18,	10,	2,
59,	51,	43,	35,	27,	19,	11,	3,
60,	52,	44,	36,	63,	55,	47,	39,
31,	23,	15,	7,	62,	54,	46,	38,
30,	22,	14,	6,	61,	53,	45,	37,
29,	21,	13,	5,	28,	20,	12,	4

Для отримання підключа відбувається циклічний зсув (величина і напрям якого фіксовані для кожного раунду) половинок попереднього ключа (табл. 2.3) [1].

Таблиця 2.3 – Число бітів циклічного зсуву C_i і D_i

Раунд	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Отримані півблоки об’єднуються в 56-бітову послідовність, з якої за допомогою постійної перестановки ущільнення вибирають 48 бітів (рис. 2.3) [1].

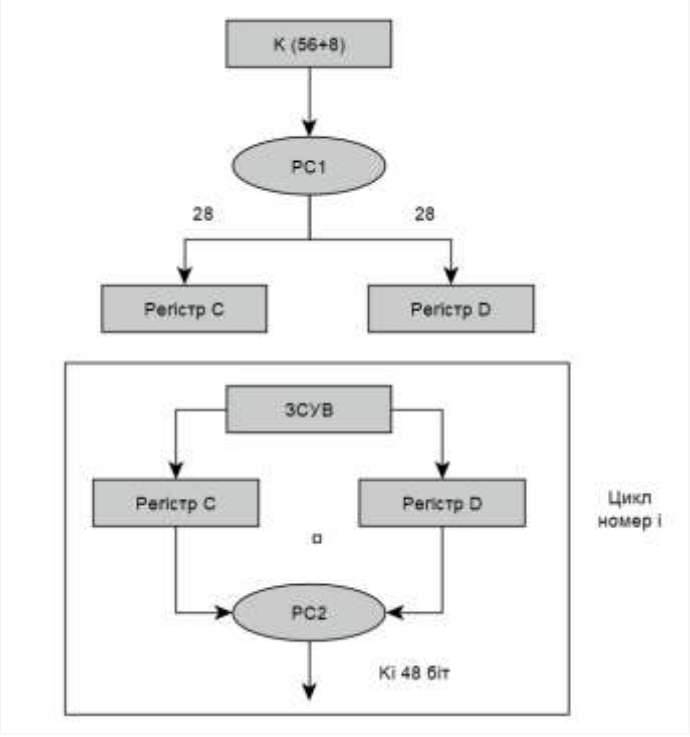


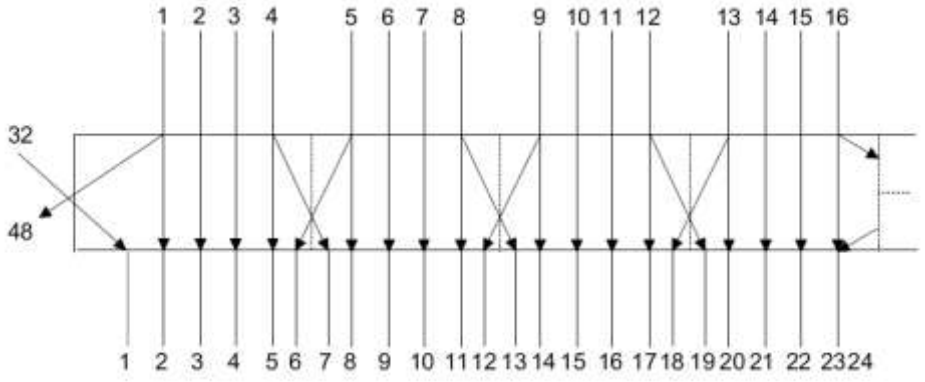
Рисунок 2.3 – Процедура формування підключів

Отже, перестановка з ущільненням передбачає не тільки вибір підмножини бітів, але й зміну їхнього порядку (табл. 2.4) [17].

Таблиця 2.4 – Перетворення PC-2

14,	17,	11,	24,	1,	5,	3,	28,
15,	6,	21,	10,	23,	19,	11,	4,
26,	8,	16,	7,	27,	20,	13,	2,
41,	52,	31,	37,	47,	55,	30,	40,
51,	45,	33,	48,	44,	49,	39,	56,
34,	53,	46,	42,	50,	36,	29,	32

На кожній ітерації алгоритму півблок даних R_{j-1}, $j = \overline{1,15}$ зазнає розширення до 48-бітового блока за допомогою перестановки, визначеної функцією E (рис. 2.4).



32,	1,	2,	3,	4,	5,	4,	5,	6,	7,	8,	9,
8,	9,	10,	11,	12,	13,	12,	13,	14,	15,	16,	17,
16,	17,	18,	19,	20,	21,	20,	21,	22,	23,	24,	25,
24,	25,	26,	27,	28,	29,	28,	29,	30,	31,	32,	1

Рисунок 2.4 – Перестановка розширення E

Наступним кроком є операція XOR між правим півблоком та раундовим ключем (рис. 2.5).

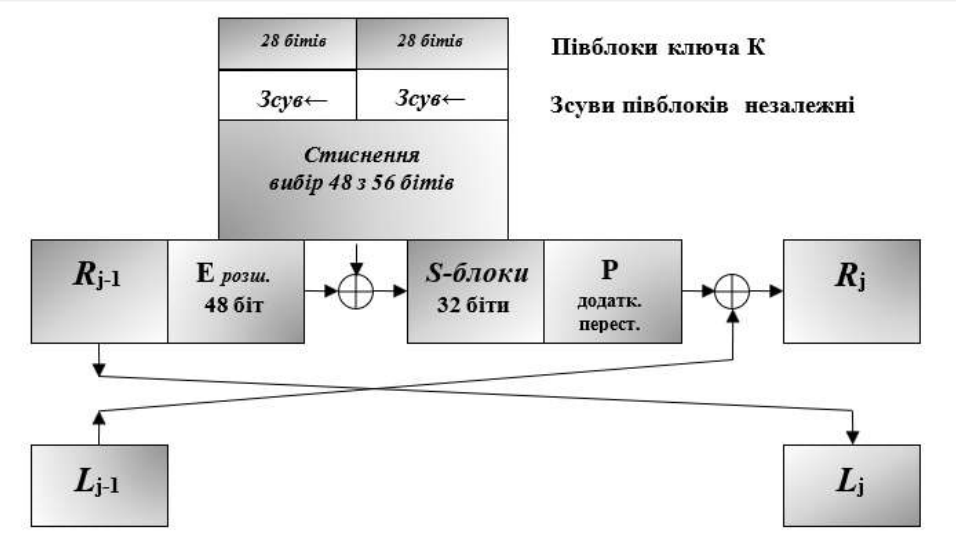


Рисунок 2.5 – Схема перетворень одного раунду

Вихідні 48 бітів (рис. 2.6) розбиваються на 8 груп по 6 бітів. Кожна група подається на вхід відповідного S-блока, який виконує нелінійну заміну, перетворюючи 6 бітів на 4. Результати роботи всіх S-блоків об'єднуються в новий 32-бітовий блок (дод. В) [17].

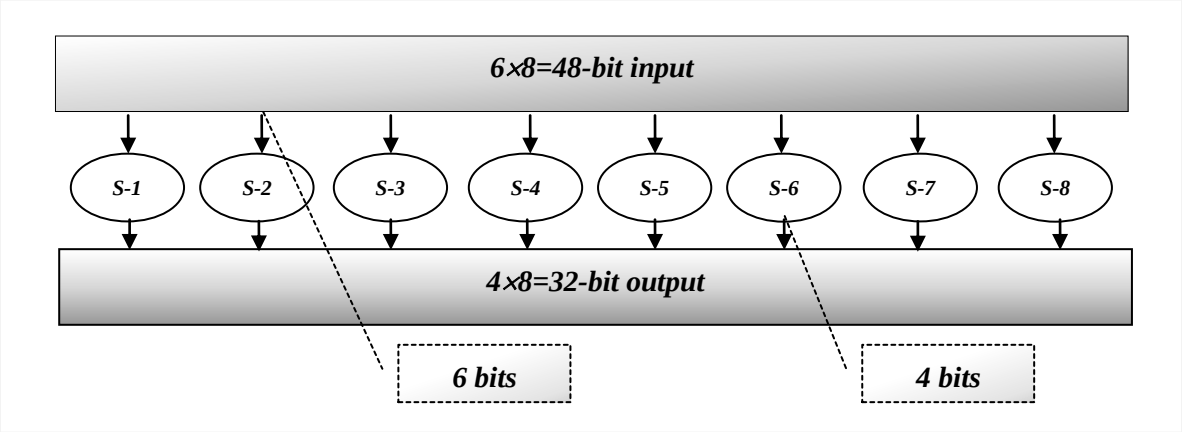


Рисунок 2.6 – Схема реалізації замін (S-блоків)

32 біти, отримані на попередньому етапі, перемішуються за схемою, заданою таблицею 2.5, і формують правий півблок R_j наступного блока.

Таблиця 2.5 – Перестановка за допомогою P-блоків

16,	7,	20,	21,	29,	12,	28,	17,
1,	15,	23,	26,	5,	18,	31,	10,
2,	8,	24,	14,	32,	27,	3,	9,
19,	13,	30,	6,	22,	11,	4,	25

Результат перестановки об'єднується з лівою половиною початкового 64-бітового блока за допомогою XOR, а потім ліва та права половини міняються місцями. Завершує алгоритм фінальна перестановка IP^{-1}, яка є інверсією початкової перестановки (табл. 2.6) [1].

Таблиця 2.6 – Перестановка IP^{-1}

40,	8,	48,	16,	56,	24,	64,	32,
39,	7,	47,	15,	55,	23,	63,	31,
38,	6,	46,	14,	54,	22,	62,	30,
37,	5,	45,	13,	53,	21,	61,	29,
36,	4,	44,	12,	52,	20,	60,	28,
35,	3,	43,	11,	51,	19,	59,	27,
34,	2,	42,	10,	50,	18,	58,	26,
33,	1,	41,	9,	49,	17,	57,	25

Існує чотири режими роботи алгоритму DES: електронна кодова книга (ECB), зчеплення шифрованих блоків (CBC), зворотний зв'язок з шифром (CFB), зворотний зв'язок за виходом (OFB) [18].

Основним недоліком криптоалгоритму DES є невелика довжина ключа, тому як альтернативу було розроблено алгоритм шифрування Triple DES, який тричі до кожного блока даних застосовує алгоритм DES. При цьому, 64-бітовий блок шифрується першим ключем, потім розшифровується другим ключем і знову зашифровується за допомогою третього ключа [9]. Проте даний шифр не отримав широкого розповсюдження через повільну швидкість роботи.

2.2 Удосконалений американський стандарт шифрування даних Rijndael (AES)

AES розшифровується як Advanced Encryption Standard і є широко використовуваним алгоритмом симетричного шифрування [19]. Переважно використовується для шифрування та захисту електронних даних. Його використовували як заміну DES (стандарту шифрування даних), оскільки він набагато швидший і кращий, ніж DES.

В 1997 році Національний інститут стандартів і технологій (NIST) США оголосив конкурс на створення нового стандарту блокового шифрування замість DES. Було розглянуто 15 заявок з різних науково-дослідних інститутів всього світу. В фінал вийшли п'ять блокових шифрів: MARS, RC6, Rijndael, Serpent та Twofish. Всі ці алгоритми були визнані криптографічно стійкими. Після підведення проміжних підсумків залишилося п'ять фіналістів, з яких в жовтні 2000 року для затвердження як стандарт AES (Advanced Encryption Standard) був вибраний криптографічний алгоритм Rijndael (Рендел) [20].

AES – це симетричний блоковий алгоритм, що використовує один секретний ключ для шифрування та розшифрування даних. Відправник і одержувач мають знати і використовувати той самий секретний ключ шифрування. Це відрізняє AES від асиметричних алгоритмів, де для шифрування та розшифрування даних використовуються різні ключі. Крім того, AES розбиває повідомлення на менші блоки та шифрує ці блоки, щоб перетворити повідомлення відкритого тексту в незрозумілу форму, яка називається зашифрованим текстом.

AES містить три блокові шифри або криптографічні ключі [18]:

– AES-128 використовує 128-бітовий ключ для шифрування та розшифрування блоків повідомлень;

– AES-192 використовує 192-бітовий ключ;

– AES-256 використовує 256-бітовий ключ.

Кожен шифр шифрує і розшифровує дані в блоках по 128 біт, використовуючи криптографічні ключі 128, 192 і 256 біт відповідно. 128-, 192- та 256-бітові ключі проходять 10, 12 і 14 раундів шифрування відповідно. Раунд (*Round*) складається з кількох етапів обробки, охоплюючи заміну, транспонування та змішування вхідного тексту для перетворення його на кінцевий вихідний зашифрований текст. Чим більше раундів, тим важче стає зламати шифрування, і тим безпечніша вихідна інформація.

Блокові шифри AES-128, AES-192 та AES-256 відрізняються за трьома параметрами:

1) довжиною ключа;

2) кількістю раундів, яка визначає розмір потрібного розкладу ключа;

3) специфікацією функції розширення раундових ключів.

Для кожного алгоритму кількість раундів позначається через N_r, яке є функцією N_k і N_b, а кількість 32-бітових слів, які складають шифроключ, – через N_k. Кількість слів у стані (*State*) позначається N_b для Rijndael у загальному випадку; у цьому стандарті $N_b = 4$. Відповідні значення N_k, N_b та N_r наведено у таблиці 2.7 [18].

Таблиця 2.7 – Основні параметри AES

	Довжина ключа N_k (bits)		Розмір блока N_b bits)		Кількість раундів N_r
AES-128	4	128	4	128	10
AES-192	6	192	4	128	12
AES-256	8	256	4	128	14

В AES відкритий текст обробляється за допомогою раундових ключів шифрування і розшифрування. Додаткові два раунди потрібні для кожних додаткових 64 біт у ключі (для 192 і 256 біт). Ключі перетворюються в матрицю 4*4, де кожен байт подано у вигляді шістнадцяткового числа.

На рисунку 2.7 показано основний процес шифрування за допомогою 128-бітового ключа в AES. Для розшифрування цей процес виконується у зворотному порядку.

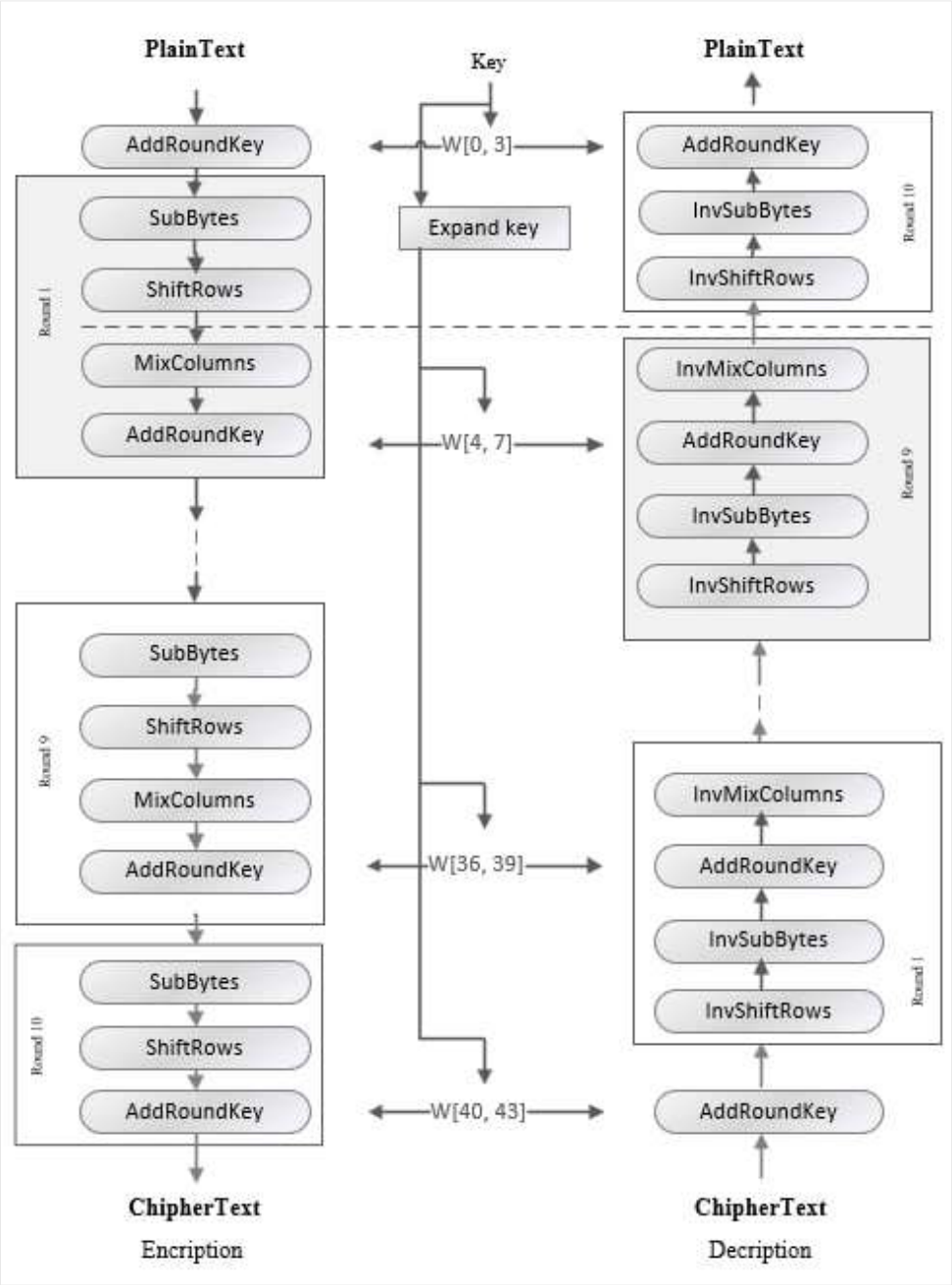


Рисунок 2.7 – Основна структурна схема AES

У процесі шифрування вхідні дані (відкритий текст) розбиваються на 128-бітові блоки. Блоки даних подано у вигляді матриць основних стовпців розміром 4 байти × 4 байти, які називаються станами (*States*) [19].

16-байтовий відкритий текст розглядається як двовимірний масив байтів ($S = S_{0,0}, S_{1,0}, ..., S_{3,3}$), як показано в таблиці 2.8, у вигляді матриці 4 × 4. Якщо довжина блока 192 біт, то стан джерела задається матрицею 6 × 4 (6 стовпців і 4 рядки), де S_{ij} – байт повідомлення. Якщо довжина блока 256 біт, то стан джерела задається матрицею 8 × 4 (8 стовпців і 4 рядки).

Таблиця 2.8 – Матриця станів (*States*)

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Джерело ключів також задається станом, послідовність зчитується також по стовпцях (табл. 2.9).

Таблиця 2.9 – Матриця ключів

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Кожен стовпець матриці ключів утворює слово w_i. Отже, ключ шифру – це чотири слова w_0, w_1, w_2, w_3, де $w_0 = k_{0,0}k_{1,0}k_{2,0}\,k_{3,0}$, $w1 = k0{,}1k1{,}1k2{,}1\,k3{,}1$ тощо.

Застосувавши алгоритм із цих слів створюється послідовність із 44 слів, кожне по 32 біти: $w_0, w_1, \ldots\ w_{43}$ [19].

У кожному раунді шифрування застосовують чотири слова цієї послідовності, що виконують роль раундового ключа (*RoundKey*) (див. рис. 2.7). Довжина раундового ключа дорівнює довжині блока N_b.

AES перетворює байти, стовпці та рядки цього масиву, породжуючи шифротекст.

Для перетворення *State* в AES використовується підстановочно-перестановочна мережа SPN, показана на рис. 2.8, з 10 раундами для 128-бітових ключів, 12 – для 192-бітових і 14 – для 256-бітових.

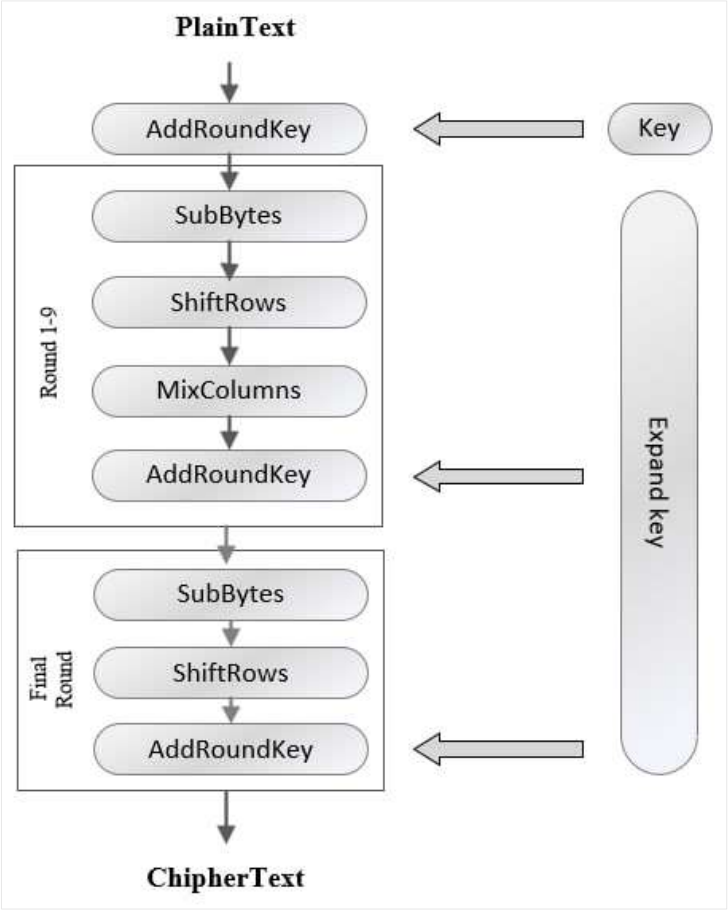


Рисунок 2.8 – Схема раундів криптоалгоритму AES

На рисунку 2.8 показано всі чотири блоки, що становлять один раунд AES. Всі раунди, окрім останнього, є послідовностями операцій SubBytes, ShiftRows, MixColumns і AddRoundKey.

AddRoundKey. Перед першим раундом застосовується операція XOR до ключа раунду і внутрішнього стану: *Round(State, RoundKey*).

SubBytes. Замінює кожен байт ($S_{0,0}$, $S_{1,0}$ ..., $S_{3,3}$) іншим байтом згідно з S-Box (рис. 2.9).

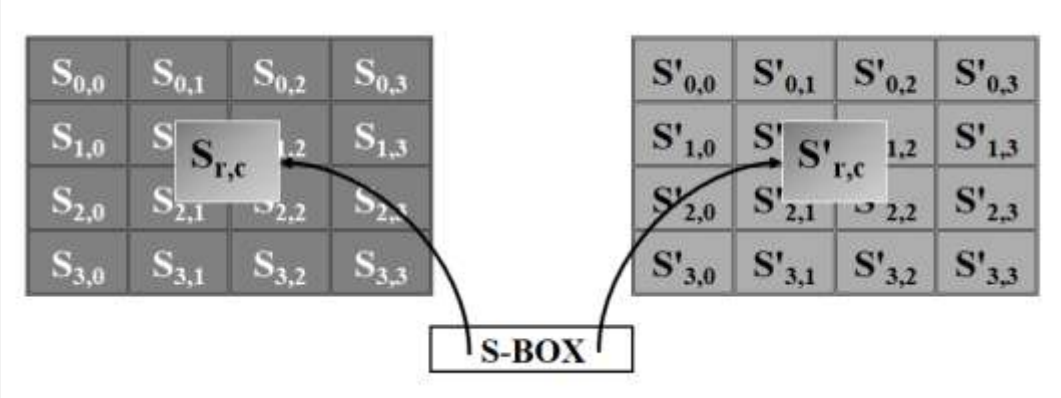


Рисунок 2.9 – Перетворення SubBytes

S-Box являє собою таблицю замін байтів в шістнадцятковій системі з 256 елементів (табл. 2.10) [19].

Таблиця 2.10 – Таблиця замін байтів S-Box

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	1	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	4	C7	23	C3	18	96	5	9A	7	12	80	E2	EB	27	B2	75
4	9	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	0	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	2	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	6	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	8
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	3	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Для перетворення InvSubBytes застосовується таблиця InvS-Box оберненої заміни, інвертована S-Box (дод. Г).

ShiftRows. Застосовується до рядків матриці *States*. Циклічний зсув *i*-го рядка на *i*-ій позицій, де *i* змінюється від 0 до 3 (рис. 2.10).

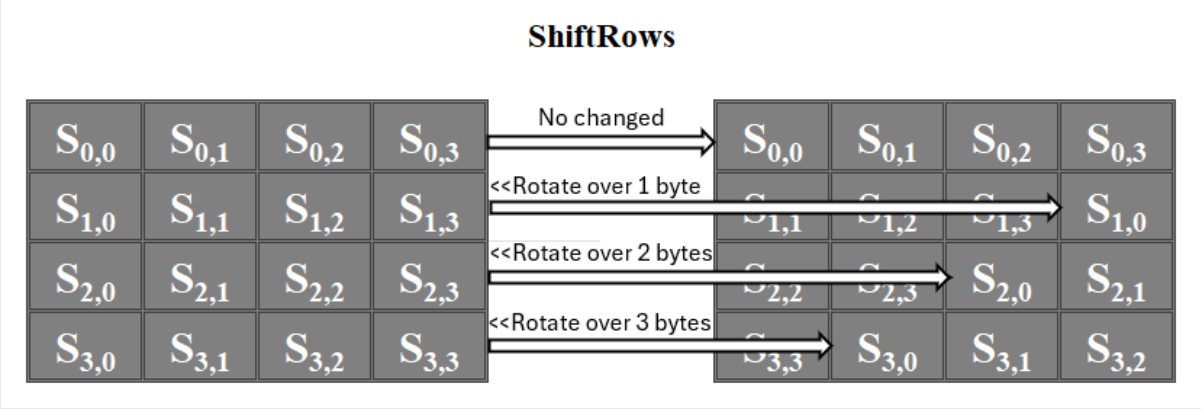


Рисунок 2.10 – Перетворення ShiftRows

MixColumns. Застосування лінійного перетворення до кожного з чотирьох стовпців стану. Тобто, стовпці *State* розглядають як многочлен над полем $GF(2^8)$ та множать за модулем $x^4 + 1$ на фіксований многочлен $c(x) = 3x^3 + x^2 + x + 2$ [20].

Після кроку ShiftRows створюється новий стовпець стану шляхом змішування чотирьох байтів кожного стовпця. Перетворення MixColumn,

насправді, поєднує кожен стовпець стану шляхом виконання множення матриці з фіксованою квадратною матрицею розміру 4×4,

де $$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}.$$

У разі табличного перетворення MixColumns кожний стовпець подається у вигляді полінома не вище третього степеня з коефіцієнтами над полем GF(2^8).

На рисунку 2.11 продемонстровано перетворення MixColumns.

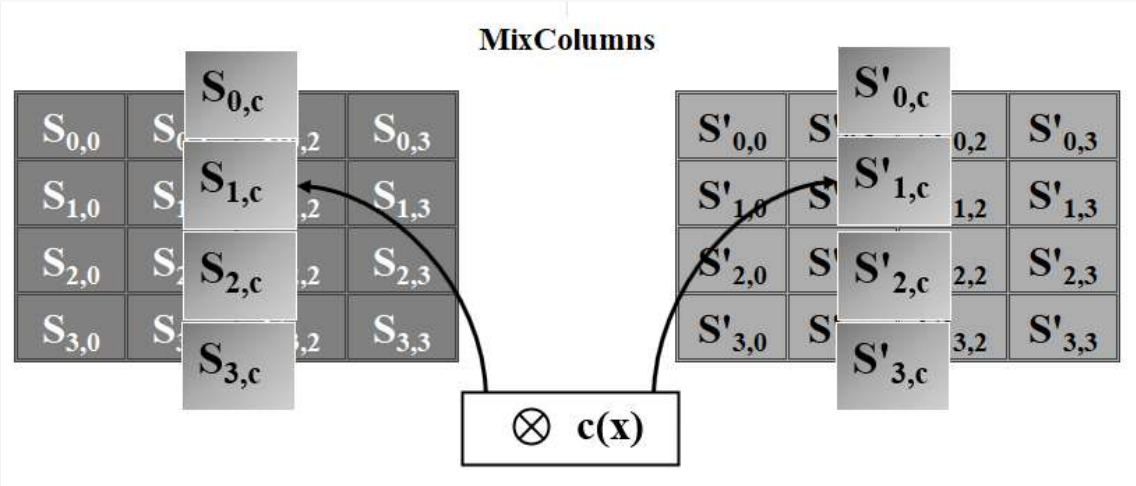


Рисунок 2.11 – Перетворення MixColumns

Операція MixColumns є основним джерелом дифузії в Rijndael.

AddRoundKey. Це перетворення стану, у якому раундовий ключ поєднується зі *State* з застосуванням побітової операції XOR. Кожен раундовий ключ складається з чотирьох слів із розкладу ключів, кожне з яких поєднується зі стовпцем стану так:

$$\left[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}\right] = \left[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}\right] \otimes \left[w_{(4*r+c)}\right], \qquad (2.3)$$

де r – значення в діапазоні $0 \le r \le N_r$,

$w[i]$ – масив ключових слів розкладу.

Перетворення AddRoundKey в процесі шифрування викликається $N_r + 1$ разів – один раз перед першим застосуванням функції розширення ключів і один раз протягом кожного з N_r раундів, коли $1 \le r \le N_r$.

Перетворення AddRoundKey продемонстровано на рисунку 2.12.

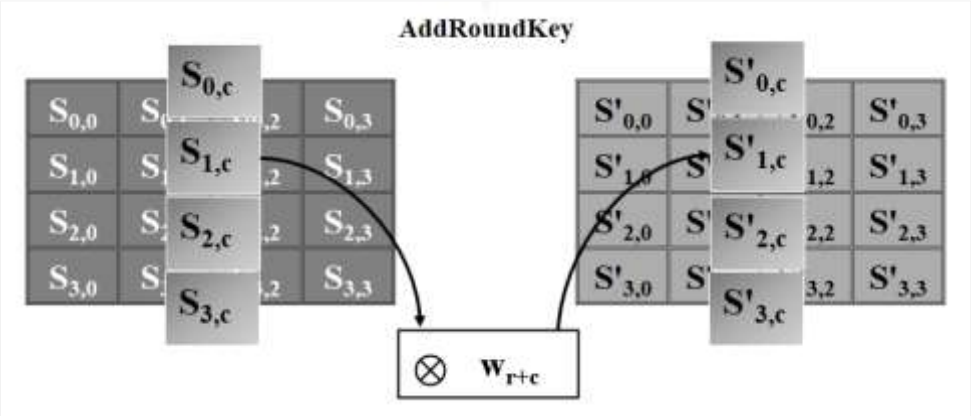


Рисунок 2.12 – Перетворення AddRoundKey

Перетворення *Expand Key* – реалізація алгоритму розширення ключа. В результаті розширення з одного 16-байтового ключа створюються 11 раундових ключів ($K_0, K_1, \ldots, K_{10}$) завдовжки 16 байт кожен, для чого використовується той самий S-блок, що й у SubBytes, і комбінація операцій XOR. Якщо зловмиснику стане відомим будь-який ключ раунду K_i, то це дасть можливість визначити ключі всіх раундів і також головний ключ, застосувавши зворотний алгоритм.

Операції виконуються над 32-розрядними словами $w[i]$. Розширення ключа показано на рисунку 2.13 для 128-бітових ключів. Початкові слова w_0, w_1, w_2, w_3 є немодифікованим ключем шифрування, який також є ключем першого раунду [19].

Отже, кожен новий раундовий ключ залежить від попереднього (рис. 2.13).

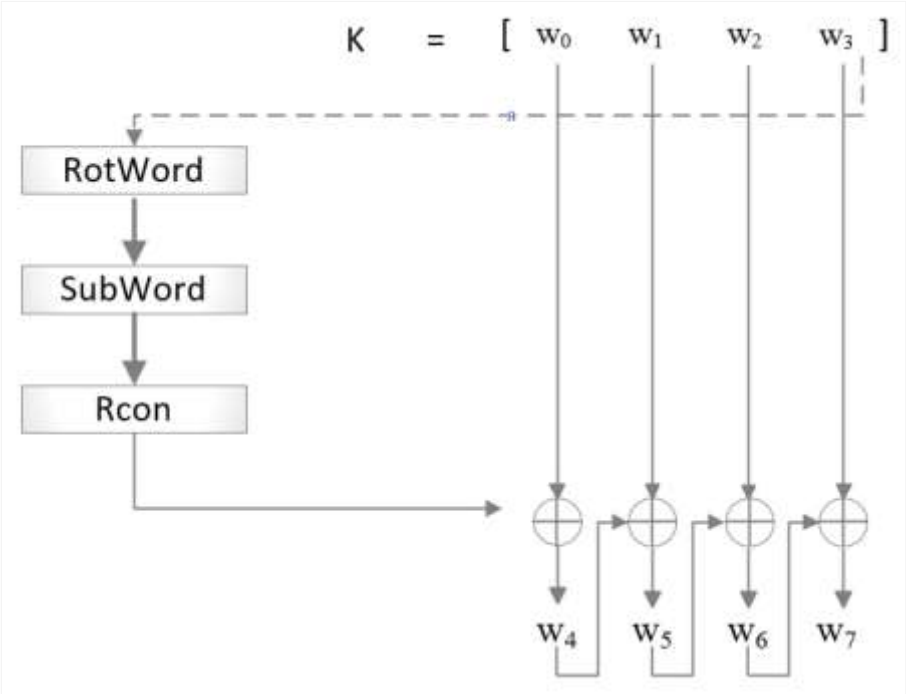


Рисунок 2.13 – Розширення ключа

RotWord. Виконує однобайтовий циклічний зсув вліво у слові w_3, яке подано у вигляді послідовності $[a_0, a_1, a_2, a_3]$

$$RotWord([a_0, a_1, a_2, a_3]) = w'_3\ [a_1, a_2, a_3, a_0]. \quad (2.4)$$

У разі виконання перетворення *SubWord* до кожного байта w'_3 застосовується операція заміни байтів *SubBytes* згідно з таблицею 2.10.

$$SubWord([a_0, a_1, a_2, a_3]) = [SB(a_0), SB(a_1), SB(a_2), SB(a_3)]. \quad (2.5)$$

До отриманого результату додається раундова константа R_{con} за модулем 2 (табл. 2.11) [19].

Таблиця 2.11 – Масив раундових констант R_{con}

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Слова $w_4 \ldots w_{43}$ отримують за основною формулою

$$w_i = w_{i-N_k} \oplus w_{i-1}, \quad (2.6)$$

За винятком, якщо $i\ mod\ 4 = 0$, тоді застосовується формула

$$w_i = w_{i-N_k} \oplus g_{i-1}, \quad (2.7)$$

де g_i є результатом застосування до слова w_i нелінійного перетворення

$$g_{i-1} = SubWord\big(RotWord(w_i)\big) \oplus R_{con}(i/N_k). \quad (2.8)$$

Отримані так слова w_i групуються в послідовності ключових елементів довжини, що дорівнює розміру шифрованого блока, для використання на раундах шифрування.

Шифр Rijndael побудований на базі прямих перетворень. Як і для всіх подібних алгоритмів, обернене перетворення будується з обернених кроків прямого перетворення, вживаних в зворотному порядку. Процедури зашифрування і розшифрування алгоритмічно ідентичні і розрізняються лише в деталях:

– в оберненому перетворенні використовується вектор замін, обернений в операційному значенні вектору замін прямого перетворення;

– в оберненому перетворенні число байтів, на які зсувається кожний рядок матриці даних в операції порядкового байтового зсуву, інше;

– в оберненому перетворенні в кроці матричного множення блок даних множиться зліва на матрицю

$$M^{-1} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix},$$

обернену тій, що використовується при прямому перетворенні;

– в оберненому перетворенні ключові елементи використовуються в оберненому порядку, і, крім того, всі елементи, за винятком першого і останнього, мають бути помножені зліва на матрицю M^{-1} [1].

Отже, алгоритм Rijndael (AES) дійсно демонструє високу гнучкість, дозволяючи ефективну реалізацію як в апаратних, так і в програмних середовищах.

2.3 Національний стандарт шифрування даних ДСТУ 7624:2014

Симетричний блоковий шифр (СБШ) «Калина» був розроблений українськими провідними науковцями у співпраці з Державною службою спеціального зв'язку та захисту інформації України. За результатами проведення відкритого національного конкурсу симетричних блокових криптоалгоритмів для поступової заміни міждержавного стандарту ДСТУ ГОСТ 28147:2009 було обрано алгоритм «Калина», який став основою для національного стандарту України ДСТУ 7624:2014 [21].

В СБШ «Калина» використані криптографічні перетворення відповідають сучасним вимогам криптографічної стійкості та швидкості. У процесі розробки алгоритму було враховано існуючі та потенційні загрози.

Алгоритм спроєктований на основі SP-мережі (AES), підтримує розмір блока та довжину ключа від 128 до 512 бітів. Довжина ключа збігається з розміром блока чи удвічі більше за нього (табл. 2.12) [22].

Таблиця 2.12 – Основні параметри СБШ «Калина»

Калина-l_b/l_k	Довжина ключа N_k l_k (bits)		Розмір блока N_b l_b (bits)		Кількість раундів N_r
Калина-128/128	2	128	2	128	10
Калина-128/256	4	256			14
Калина-256/256	4	256	4	256	14
Калина-256/512	8	512			18
Калина-512/512	8	512	8	512	18

ДСТУ 7624:2014 визначає 10 режимів роботи для забезпечення конфіденційності та цілісності відповідно до міжнародного стандарту ISO/IEC 10116:2006 (табл. 2.13) [21].

Таблиця 2.13 – Режими роботи СБШ «Калина»

Ч.ч.	Назва режиму	Позна-чення	Послуга безпеки
1	Проста заміна (базове перетворення)	ECB	Конфіденційність
2	Гамування	CTR	Конфіденційність
3	Гамування зі зворотним зв’язком за шифротекстом	CFB	Конфіденційність
4	Вироблення імітовставки	CMAC	Цілісність
5	Зчеплення шифроблоків	CBC	Конфіденційність
6	Гамування зі зворотним зв’язком за шифрограмою	OFB	Конфіденційність
7	Вибіркове гамування з прискореним виробленням імітовставки	GCM, GMAC	Конфіденційність і цілісність (GCM), тільки цілісність (GMAC)
8	Вироблення імітовставки та гамування	CCM	Цілісність і конфіденційність
9	Індексована заміна	XTS	Конфіденційність
10	Захист ключових даних	KW	Конфіденційність і цілісність

Основна структурна схема алгоритму «Калина» в режимах зашифрування і розшифрування наведена на рисунку 2.14.

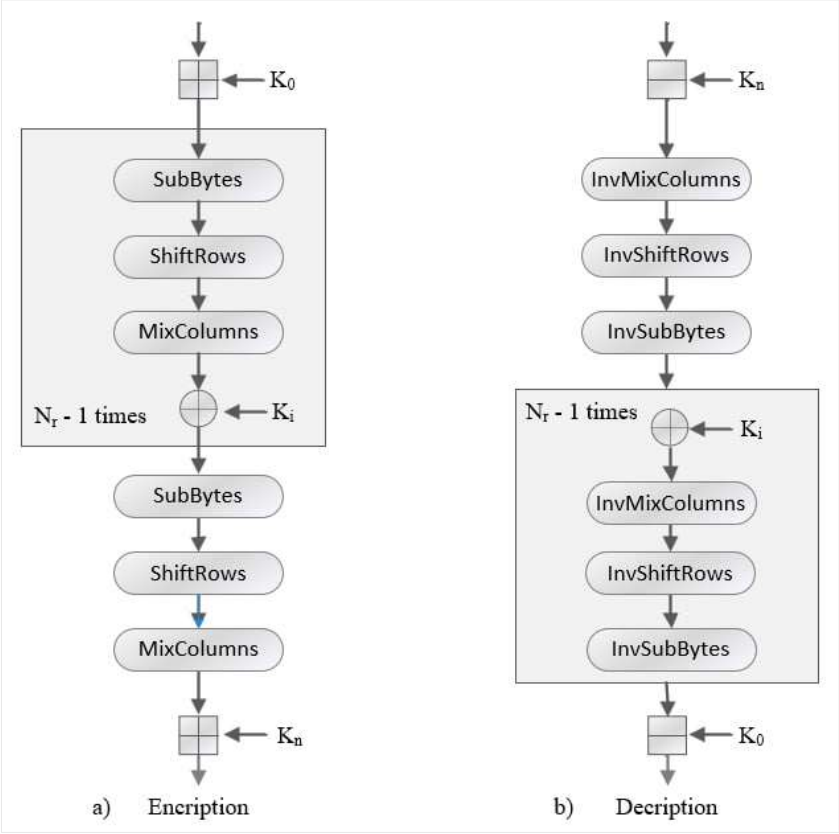


Рисунок – 2.14 Основна структурна схема «Калина» в режимах шифрування і розшифрування

В алгоритмі використовуються такі операції:

– арифметичні додавання (⊞) та віднімання (⊟) за модулем 2^{64};

– додавання за модулем 2 (⊕) (XOR);

– нелінійна заміна (SubBytes, InvSubBytes);

– циклічний зсув рядків (ShiftRows, InvShiftRows);

– лінійне перетворення (MixColumns, InvMixColumns) [23].

Якщо відкритий текст визначено у вигляді послідовності байтів $in_1, in_2, \dots\ in_{15},$ а шифротекст – $out_1, out_2, \dots\ out_{15}$, то заповнення матриці стану (*State*) розміром 8×2 для довжини блока 128 у процесі шифрування – $S_{0,0}, S_{1,0}, \dots\ S_{7,1}$ (рис. 2.15).

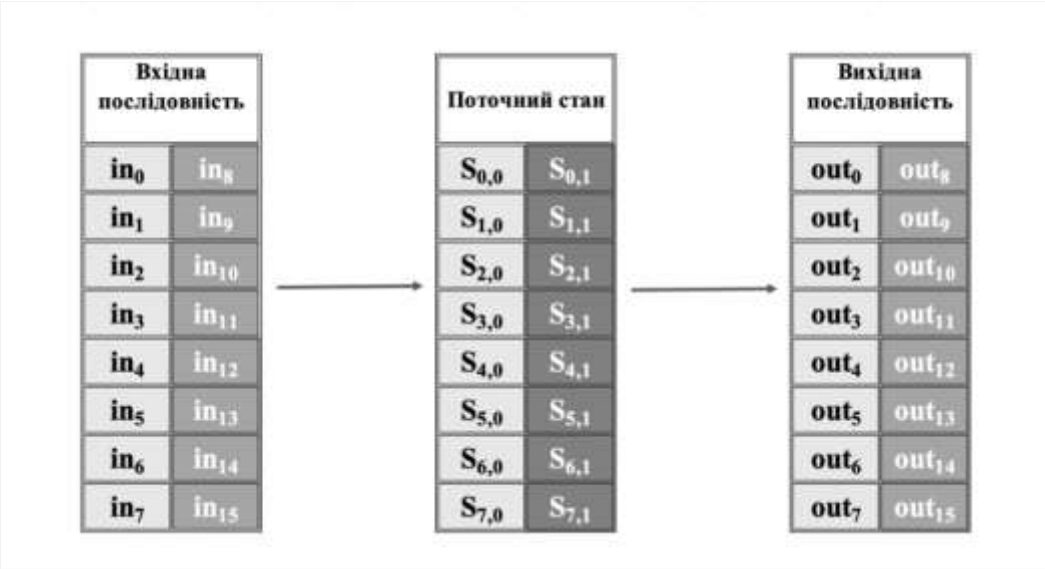


Рисунок 2.15 – Заповнення матриці стану для довжини блока 128 біт

Поточний стан шифру (*State)* подано у вигляді матриці розмірністю $8 \times N_b$ (вісім рядків по N_b байт) [23], що являють собою елементи поля $GF(2^8)$: *State* = $S_{i,j}$, де $i = 0, 1, \dots 7,\ j = 0, 1, \dots N_b - 1$.

Наприклад, матриця стану при $N_b = 4$ (256 біт)

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\ S_{4,0} & S_{4,1} & S_{4,2} & S_{4,3} \\ S_{5,0} & S_{5,1} & S_{5,2} & S_{5,3} \\ S_{6,0} & S_{6,1} & S_{6,2} & S_{6,3} \\ S_{7,0} & S_{7,1} & S_{7,2} & S_{7,3} \end{pmatrix}.$$

Ключ шифру подано у вигляді матриці байтів розмірністю $8 \times N_b$. Отримаємо матрицю ключа шифру при $N_k = 4$ (256 біт)

$$\begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ k_{4,0} & k_{4,1} & k_{4,2} & k_{4,3} \\ k_{5,0} & k_{5,1} & k_{5,2} & k_{5,3} \\ k_{6,0} & k_{6,1} & k_{6,2} & k_{6,3} \\ k_{7,0} & k_{7,1} & k_{7,2} & k_{7,3} \end{pmatrix}.$$

Операції арифметичного додавання ($\boxplus$) та віднімання ($\boxminus$) за модулем 2^{64}, у процесі шифрування та розшифрування відповідно, реалізують додавання або віднімання стовпців матриці стану *State* і стовпців раундового підключа за модулем 2^{64} [23].

У разі операції додавання менші значущі байти мають менші індекси, тобто числа в стовпцях подані [24] у форматі little endian (від молодшого до старшого байта). Отриманий результат записується на місце першого аргументу, наприклад, для станів розміром 256 біт

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\ S_{4,0} & S_{4,1} & S_{4,2} & S_{4,3} \\ S_{5,0} & S_{5,1} & S_{5,2} & S_{5,3} \\ S_{6,0} & S_{6,1} & S_{6,2} & S_{6,3} \\ S_{7,0} & S_{7,1} & S_{7,2} & S_{7,3} \end{pmatrix} \boxplus \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ k_{4,0} & k_{4,1} & k_{4,2} & k_{4,3} \\ k_{5,0} & k_{5,1} & k_{5,2} & k_{5,3} \\ k_{6,0} & k_{6,1} & k_{6,2} & k_{6,3} \\ k_{7,0} & k_{7,1} & k_{7,2} & k_{7,3} \end{pmatrix} = \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & k_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} \\ b_{5,0} & b_{5,1} & b_{5,2} & b_{5,3} \\ b_{6,0} & b_{6,1} & b_{6,2} & b_{6,3} \\ b_{7,0} & b_{7,1} & b_{7,2} & b_{7,3} \end{pmatrix}.$$

Перетворення SubBytes, операція нелінійної заміни, виконує заміну кожного байта [24] матриці стану відповідно до заданих таблиць підстановки «байт-в-байт» $\pi_0, \ldots \pi_3$, а для перетворення InvSubBytes – відповідно до обернених таблиць підстановки ${}_{-1}\pi_0, \ldots, {}_{-1}\pi_3$ (дод. Ґ). Для байтів одного рядка матриці стану застосовується одна й та сама підстановка. Наприклад, для всіх елементів 1-го ($S_{1,i}$) та 5-го рядків ($S_{5,i}$) застосовують таблицю підстановки π_1 (табл. 2.14).

Таблиця 2.14 – Порядок підстановки «байт-в-байт» $\pi_0, \ldots \pi_3$

0-ряд	$S_{0,i}$	
1-ряд	$S_{1,i}$	$\pi_1^{\pi_0}$
2-ряд	$S_{2,i}$	π_2
3-ряд	$S_{3,i}$	π_3
4-ряд	$S_{4,i}$	π_0
5-ряд	$S_{5,i}$	π_1
6-ряд	$S_{6,i}$	π_2
7-ряд	$S_{7,i}$	π_3

Заміна байта для розміру блока 256 біт під час перетворення SubBytes наведена на рисунку 2.16 [23].

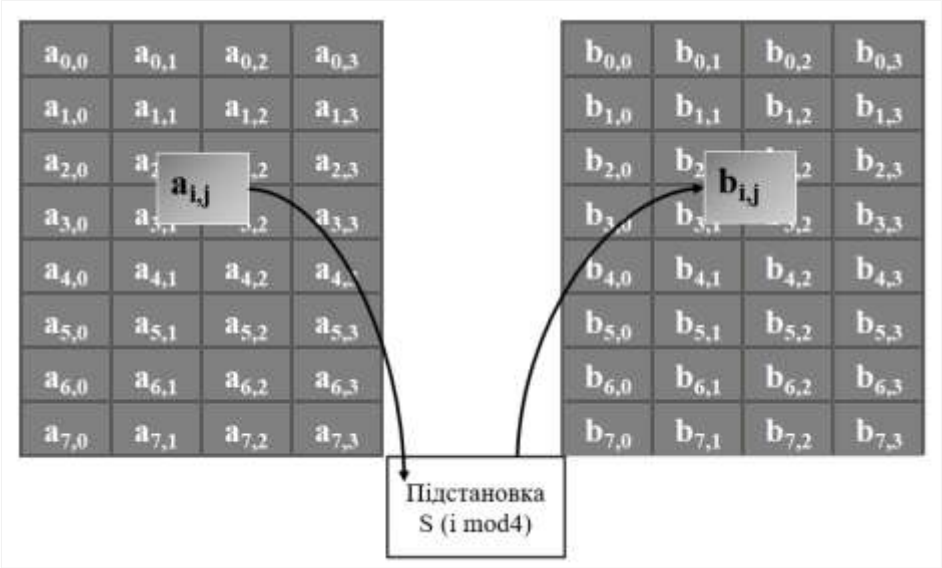


Рисунок 2.16 – Приклад підстановки байта для розміру блока 256 біт

Отже, номер таблиці підстановки визначається як індекс рядка байта за модулем 4 $s_{i,j} = S_{i \bmod 4}(s_{i,j})$ чи $s_{i,j} = inv\_S_{i \bmod 4}(s_{i,j})$ (дод. Д) [21].

У таблиці 2.15 наведено приклад підстановки байта $9B_{16}$ для таблиці π_0 [21].

Таблиця 2.15 – Приклад підстановки для таблиці π_0

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	A8	43	5F	6	6B	75	6C	59	71	DF	87	95	17	F0	D8	9
1	6D	F3	1D	CB	C9	4D	2C	AF	79	E0	97	FD	6F	4B	45	39
2	3E	DD	A3	4F	B4	B6	9A	0E	1F	BF	15	E1	49	D2	93	C6
3	92	72	9E	61	D1	63	FA	EE	F4	19	D5	AD	58	A4	BB	A1
4	DC	F2	83	37	42	E4	7A	32	9C	CC	AB	4A	8F	6E	4	27
5	2E	E7	E2	5A	96	16	23	2B	C2	65	66	0F	BC	A9	47	41
6	34	48	FC	B7	6A	88	A5	53	86	F9	5B	DB	38	7B	C3	1E
7	22	33	24	28	36	C7	B2	3B	8E	77	BA	F5	14	9F	8	55
8	9B	4C	FE	60	5C	DA	18	46	CD	7D	21	B0	3F	1B	89	FF
9	EB	84	69	3A	9D	D7	D3	70	67	40	B5	DE	5D	30	91	B1
A	78	11	1	E5	0	68	98	A0	C5	2	A6	74	2D	0B	A2	76
B	B3	BE	CE	BD	AE	E9	8A	31	1C	EC	F1	99	94	AA	F6	26
C	2F	EF	E8	8C	35	3	D4	7F	FB	5	C1	5E	90	20	3D	82
D	F7	EA	0A	0D	7E	F8	50	1A	C4	7	57	B8	3C	62	E3	C8
E	AC	52	64	10	D0	D9	13	0C	12	29	51	B9	CF	D6	73	8D
F	81	54	C0	ED	4E	44	A7	2A	85	25	E6	CA	7C	8B	56	80

Отже, рядок буде визначений старшими 4 бітами, а стовпець – молодшими 4 бітами байта в шістнадцятковому поданні. Для значення $9B_{16}$ результатом підстановки є число DE_{16}, що знаходиться на перетині 10-го рядка (індекс 9) та 12-го стовпця (індекс B).

У процесі виконання перетворень ShiftRows або InvShiftRows здійснюється зсув рядків стану вправо або вліво, відповідно, на кількість байтів, залежно від розміру блока l та значення індексу рядка i (табл. 2.16) [22].

Таблиця 2.16 – Значення циклічних зсувів рядків

Номер рядка	Значення зсуву, байти		
	Довжина блока 128 біт	Довжина блока 256 біт	Довжина блока 512 біт
0	0	0	0
1	0	0	1
2	0	1	2
3	0	1	3
4	1	2	4
5	1	2	5
6	1	3	6
7	1	3	7

Число позицій δ_i обчислюється за формулою (2.9)

$$\delta_i = \left[\frac{i \cdot l}{512}\right]. \tag{2.9}$$

На рисунку 2.17 продемонстровано порядок виконання перетворення ShiftRows для різних за розмірами блоків.

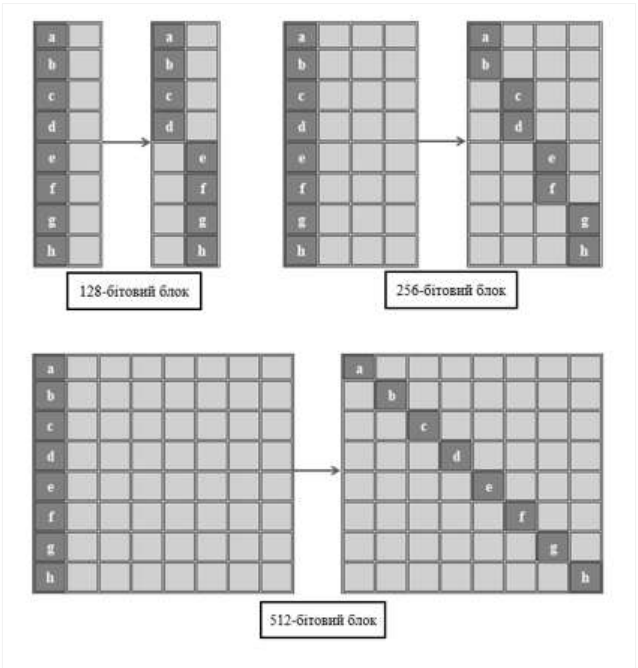


Рисунок 2.17 – Розподіл байтів 1-го стовпця при здійсненні перетворення ShiftRows

На рисунку 2.18 продемонстровано порядок виконання перетворення InvShiftRows для різних розмірів блока [21].

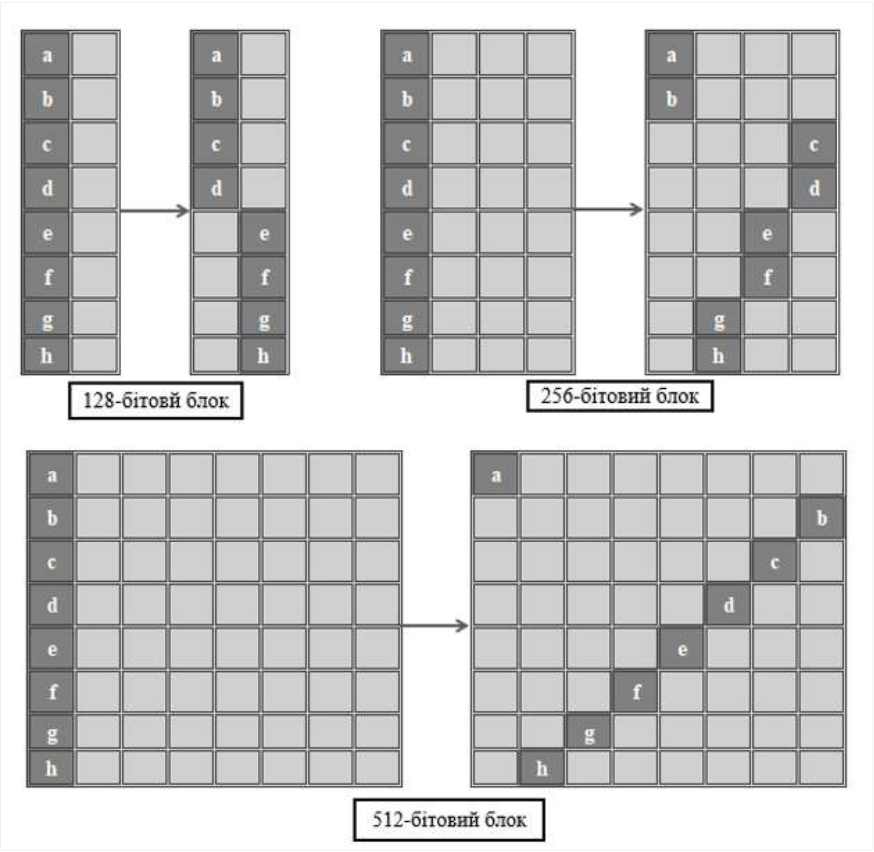


Рисунок 2.18 – Розподіл байтів 1-го стовпця при здійсненні перетворення InvShiftRows

В процесі виконання перетворення MixColumns послідовно обробляються всі стовпці поточного стану. Кожний стовпець із 8 байт подано як поліном над скінченним полем GF(2^8), що складається з суми 8 одночленів та виконується множенням цього полінома за модулем $x^8 + 1$ на фіксований многочлен C(x) [21]

$$C(x) = \{01\}x^7 + \{05\}x^6 + \{01\}x^5 + \{08\}x^4 + \{06\}x^3 + \{07\}x^2 + \{04\}x + \{01\}.$$

Ця операція рівнозначна матричному множенню над GF(2^8) початкового вектора a на фіксовану матрицю, результатом є отриманий вектор b

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 01 & 01 & 05 & 01 & 08 & 06 & 07 & 04 \\ 04 & 01 & 01 & 05 & 01 & 08 & 06 & 07 \\ 07 & 04 & 01 & 01 & 05 & 01 & 08 & 06 \\ 06 & 07 & 04 & 01 & 01 & 05 & 01 & 08 \\ 08 & 06 & 07 & 04 & 01 & 01 & 05 & 01 \\ 01 & 08 & 06 & 07 & 04 & 01 & 01 & 05 \\ 05 & 01 & 08 & 06 & 07 & 04 & 01 & 01 \\ 01 & 05 & 01 & 08 & 06 & 07 & 04 & 01 \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix}.$$

Операція множення у полі GF(2^8) виконується за допомогою незвідного полінома $M(x) = x^8 + x^4 + x^3 + x^2 + 1$.

На рисунку 2.19 зображено перетворення MixColumns для поточного стану шифру з розміром блока 256 біт [22].

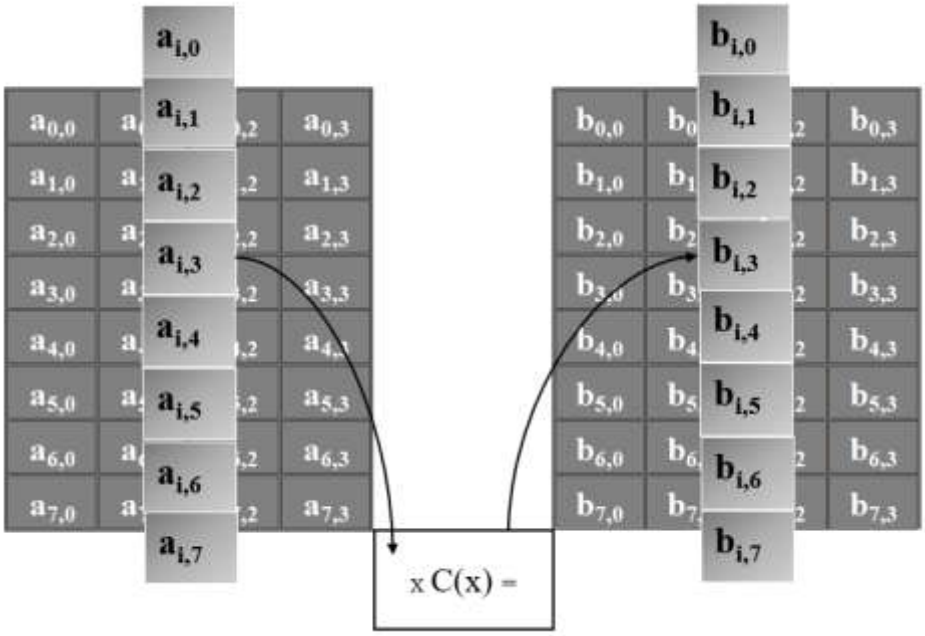


Рисунок 2.19 – Перетворення MixColumns для поточного стану шифру з розміром блока 256 біт

Додавання за модулем 2 (⊕) (XOR) раундового ключа. В процесі операції XOR виконується побітове додавання до матриці стану раундового ключа і результат записується на місце першого аргументу

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\ S_{4,0} & S_{4,1} & S_{4,2} & S_{4,3} \\ S_{5,0} & S_{5,1} & S_{5,2} & S_{5,3} \\ S_{6,0} & S_{6,1} & S_{6,2} & S_{6,3} \\ S_{7,0} & S_{7,1} & S_{7,2} & S_{7,3} \end{pmatrix} \oplus \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \\ k_{4,0} & k_{4,1} & k_{4,2} & k_{4,3} \\ k_{5,0} & k_{5,1} & k_{5,2} & k_{5,3} \\ k_{6,0} & k_{6,1} & k_{6,2} & k_{6,3} \\ k_{7,0} & k_{7,1} & k_{7,2} & k_{7,3} \end{pmatrix} = \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & k_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ b_{4,0} & b_{4,1} & b_{4,2} & b_{4,3} \\ b_{5,0} & b_{5,1} & b_{5,2} & b_{5,3} \\ b_{6,0} & b_{6,1} & b_{6,2} & b_{6,3} \\ b_{7,0} & b_{7,1} & b_{7,2} & b_{7,3} \end{pmatrix}.$$

Для матриці стану $S_{i,j}$ та раундового ключа $k_{i,j}$ результатом перетворення додавання за модулем 2 є $b_{i,j}$ (2.10)

$$b_{i,j} = S_{i,j} \oplus k_{i,j}. \qquad (2.10)$$

У процесі формування ключів використовується $N_r + 1$ раундових ключів k_i $(i = 0,1,\dots,N_r)$, кожний довжиною $64 \times N_b$ біт. Розмір шифротексту і матриці стану збігається з розміром раундового ключа.

Розгортання ключа відбувається так:

– допоміжний ключ K_t довжиною $64 \times N_b$ біт формується з ключа шифрування з використанням 3-х раундів шифрування (рис. 2.20) [22].

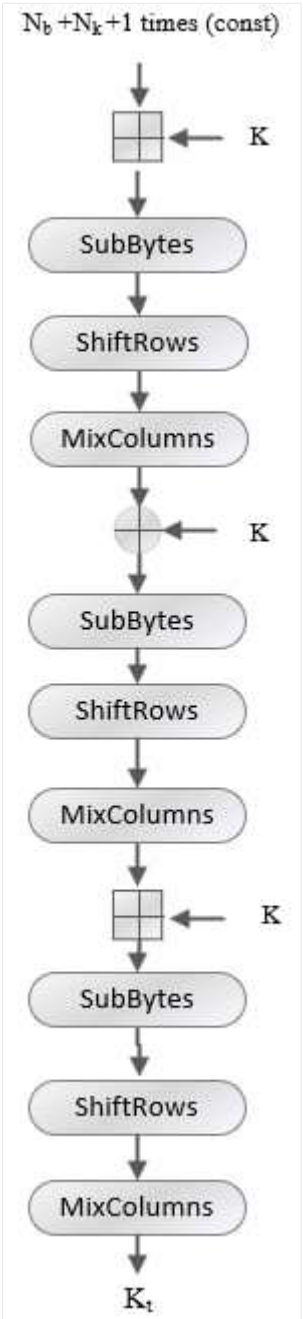


Рисунок 2.20 – Схема формування допоміжного ключа K_t

На вхід подається число $N_b + N_k + 1$ у двійковому вигляді, інші байти заповнюються нулями.

– раундові ключі з парними індексами K_{2i} довжиною $64 \times N_b$ біт формуються на основі ключа K та допоміжного ключа K_t після 2-х раундів шифрування для кожного раундового ключа [12]. Раундовим ключем є результат додавання за модулем 2^{64} допоміжного ключа K_t та змінної tmv_i (рис. 2.21) [25].

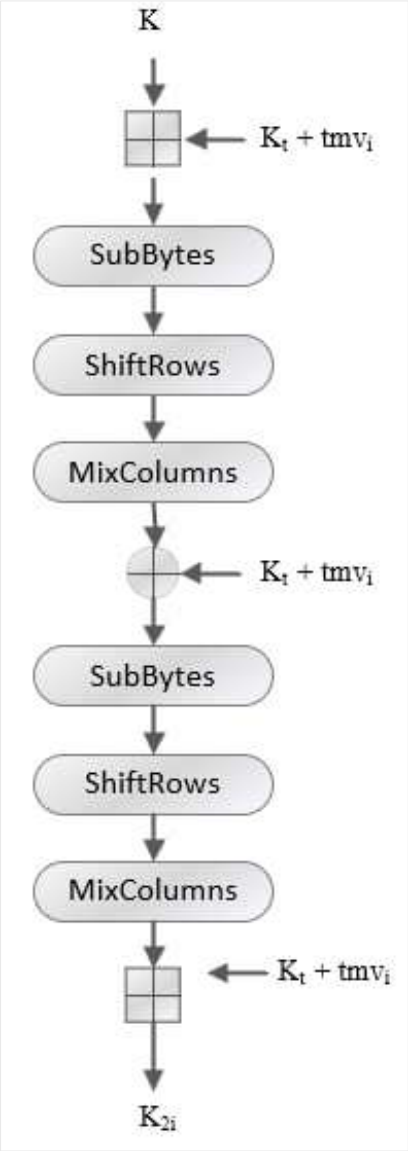


Рисунок 2.21 – Схема формування раундових ключів

– раундові ключі з непарними індексами K_{2i+1} формуються за допомогою циклічного зсуву попереднього ключа K_{2i} вліво на $2N_b + 3$ байт.

Процес розшифрування є оберненим до процесу шифрування.

Отже, криптоалгоритм «Калина», визначений у стандарті України ДСТУ 7624:2014, враховує сучасні світові тенденції у сфері криптографії та надає можливість одночасного забезпечення конфіденційності та цілісності відкритого тексту [26] з застосуванням відповідних перетворень.

2.4 Потоковий симетричний криптоалгоритм «Струмок»

Алгоритм «Струмок» (ДСТУ 8845:2019) [27] працює з 64-бітовими словами, використовуючи 256-бітовий вектор ініціалізації *IV* та ключ довжиною 256 або 512 біт. В цьому випадку байти в ключі та векторі ініціалізації записуються від старшого до молодшого біта.

Потоковий шифр «Струмок» побудовано за SNOW-2.0-подібною схемою підсумовувального генератора. Ключова відмінність полягає в тому, що SNOW-2.0 розроблений для 32-бітових систем, а «Струмок» – для більш продуктивних 64-бітових платформ. Завдяки цьому алгоритм «Струмок» генерує псевдовипадкові послідовності значно швидше. Крім того, порівняно з шифром SNOW-2.0, алгоритм «Струмок» має більші довжини секретного ключа та вектора ініціалізації, що гарантує його стійкість навіть з урахуванням розвитку квантових методів криптоаналізу [27].

Генератор «Струмок» складається з лінійного регістра зсуву (LFSR) та скінченного автомата (FSM), де відбувається нелінійне перетворення вихідної послідовності.

Структура змінної стану S_i $(i \geq 0)$ містить вісімнадцять 64-бітових елементи, кожен з яких має двокомпонентну структуру:

– шістнадцять змінних $s^{(i)}$ – комірки регістра зсуву з лінійним зворотним зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, \ldots, s_0^{(i)})$;

– два регістри скінченного автомата: $r^{(i)} = \left(r_2^{(i)}, r_1^{(i)}\right)$ [28].

Результатом є генерація ключового потоку (гами шифру), що складається з 64-бітових слів Z_i.

На рисунку 2.22 зображено схему роботи шифру «Струмок» у режимі генерації гами в момент часу i [27].

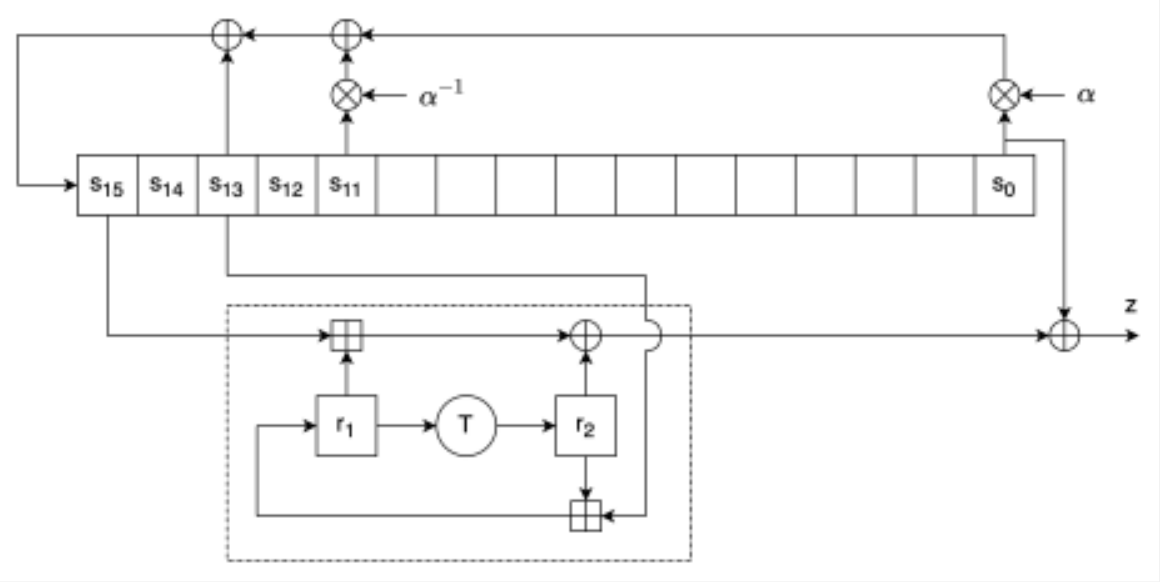


Рисунок 2.22 – Схема роботи алгоритму симетричного шифрування «Струмок»

За допомогою примітивного над полем $GF(2^{64})$ полінома формуються відводи зворотного зв'язку у LFSR: $f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha$, де α є коренем примітивного над полем $GF(2^8)$ полінома $g(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2 z^5 + \beta^{224}z^4 + \beta^{70}z^3 + \beta^2$, де β – є коренем примітивного над полем $GF(2)$ полінома $p(y) = y^8 + y^4 + y^3 + y^2 + 1$. У

результаті вкладення полів $GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024})$ лінійний регістр зсуву генерує послідовність з максимально можливим періодом $2^{1024} - 1$ [27].

В основі криптоалгоритму «Струмок» лежать такі фундаментальні функції, як *Init, Next, Strm ma FSM*. Розглянемо більш детально математичну структуру зазначених функцій.

Функція *Init* використовує 256-бітовий вектор ініціалізації IV та 256- або 512-бітовий ключ шифрування K, на основі яких обчислює початкове значення змінної стану $S_0 = \left(s^{(0)}, r^{(0)}\right)$ [28].

Ключ алгоритму «Струмок-256» складається з чотирьох 64-бітових слів $K = (K_3, K_2, K_1, K_0)$, а в режимі «Струмок-512» – з восьми 64-бітових слів $K = (K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0)$. У обох режимах послідовність слів ключа розміщена за вагомістю, де K_3 та K_7, відповідно, для 256 та 512 біт є найбільш вагомі слова, а K_0 – найменш вагомі.

Структура вектора ініціалізації для обох режимів ідентична і складається з чотирьох 64-бітових слів: $IV = (IV_3, IV_2, IV_1, IV_0)$ [27].

1. У шістнадцять комірок LFSR записують значення ключа.

У разі використання 256-бітового ключа K алгоритм передбачає виконання таких операцій:

$$s_{15}^{(-33)} = \neg K_0,\ s_{14}^{(-33)} = K_1, s_{13}^{(-33)} = \neg K_2, s_{12}^{(-33)} = K_3, s_{11}^{(-33)} = K_0,$$
$$s_{10}^{(-33)} = \neg K_1, s_9^{(-33)} = K_2, s_8^{(-33)} = K_3, s_7^{(-33)} = \neg K_0, s_6^{(-33)} = \neg K_1,$$
$$s_5^{(-33)} = K_2 \oplus IV_3, s_4^{(-33)} = K_3, s_3^{(-33)} = K_0 \oplus IV_2,$$
$$s_2^{(-33)} = K_1 \oplus IV_1, s_1^{(-33)} = K_2,\ s_0^{(-33)} = K_3 \oplus IV_0$$

Для ключа K довжиною 512 біт передбачено нижченаведені операції:

$$s_{15}^{(-33)} = K_0,\ s_{14}^{(-33)} = \neg K_1, s_{13}^{(-33)} = K_2, s_{12}^{(-33)} = K_3, s_{11}^{(-33)} = \neg K_7,$$
$$s_{10}^{(-33)} = K_5, s_9^{(-33)} = \neg K_6, s_8^{(-33)} = K_4 \oplus IV_3, s_7^{(-33)} = \neg K_0, s_6^{(-33)} = K_1,$$
$$s_5^{(-33)} = K_2 \oplus IV_2, s_4^{(-33)} = K_3, s_3^{(-33)} = K_4 \oplus IV_1,$$
$$s_2^{(-33)} = K_5, s_1^{(-33)} = K_6,\ s_0^{(-33)} = K_7 \oplus IV_0$$

2. Протягом наступних тридцяти двох тактів відбувається ініціалізація алгоритму без формування ключового потоку

$$S_{-1} = Next^{32}(S_{-33}, INIT). \qquad (2.11)$$

Протягом тридцяти двох ітерацій функція *Next* виконує потрібні дії в режимі ініціалізації *INIT*. Значення змінної стану $S_{-33} = \left(s^{(-33)}, r^{(-33)}\right)$ відповідає шістнадцятьом коміркам регістра зсуву $s^{(-33)}$, обчисленим на попередньому кроці, та двом початковим нульовим значенням регістрів зворотного зв'язку $r^{(-33)} = \left(r_2^{(-33)}, r_1^{(-33)}\right)$ скінченного автомата [27].

3. Початкове значення стану S_0 отримується шляхом застосування функції $NEXT$ до попереднього стану S_{-1}. При цьому функція NEXT працює в звичайному режимі: $S_0 = NEXT(S_{-1})$.

4. Процес завершується отриманням вихідного значення

$$S_0 = \left(s^{(0)}, r^{(0)}\right). \quad (2.12)$$

Функція *Next* оперує зі зміною поточного стану $S_i = \left(s^{(i)}, r^{(i)}\right)$ та генерує зміну наступного стану $S_i = \left(s^{(i+1)}, r^{(i+1)}\right)$. Залежно від методу виконання ітерацій вказана функція може працювати в двох режимах: звичайному та режимі ініціалізації.

1. З метою оновлення значення регістра $r_2^{(i+1)}$ скінченного автомата здійснюється нелінійна підстановка, яка ґрунтується на розрахунку значення функції T,

$$r_2^{(i+1)} = T\left(r_1^{(i)}\right). \quad (2.13)$$

2. Наступним кроком є оновлення значення регістра r_1 за формулою

$$r_1^{(i+1)} = r_2^{(i)} +_{64} S_{13}^{(i)}, \quad (2.14)$$

де операція $+_{64}$ – додавання за модулем 2^{64} [29].

3. У 15 комірках відбувається оновлення значень LFSR для усіх $j = 0, 1, \dots, 14$.

$$S_j^{(i+1)} = S_{j+1}^{(i)}. \quad (2.15)$$

4. Вміст 16-ї комірки LFSR оновлюється, причому якщо встановлено звичайний режим функції Next, то $s_{15}^{(i+1)} = s_0^{(i)} \cdot \alpha + s_{11}^{(i)} \cdot \alpha^{-1} + s_{13}^{(i)}$, у випадку режиму ініціалізації –

$$s_{15}^{(i+1)} = s_0^{(i)} \cdot \alpha + s_{11}^{(i)} \cdot \alpha^{-1} + s_{13}^{(i)} + FSM\left(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}\right). \quad (2.16)$$

5. Процес завершується отриманням вихідного значення

$$S_i = \left(s^{(i)}, r^{(i)}\right). \quad (2.17)$$

Рисунок 2.23 демонструє роботу генератора «Струмок» під час виконання функції *Next* у режимі ініціалізації.

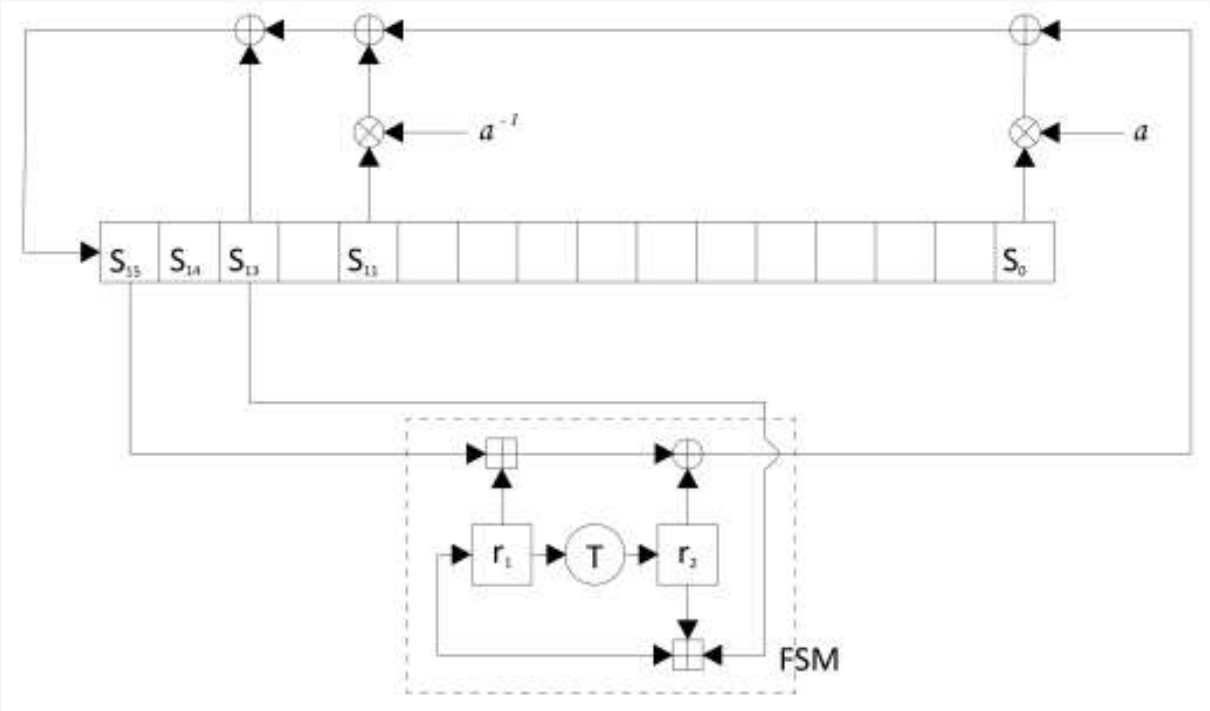


Рисунок 2.23 – Ініціалізація криптографічного алгоритму «Струмок»

Використовуючи вхідну змінну стану $S_i = \left(s^{(i)}, r^{(i)}\right)$, **функція *Strm*** формує 64-бітовий ключовий потік Z_i [29]

$$Z_i = FSM\left(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}\right) \oplus s_0^{(i)}. \quad (2.18)$$

Функція *FSM* на основі трьох вхідних 64-бітових рядків x, y та z обчислює 64-бітовий рядок $q = (x +_{64} y) \oplus z$.

Функція *T* реалізує перестановку елементів поля $GF(2^{64})$ на основі компонентів українського стандарту шифрування «Калина». Дана функція опрацьовує 64-бітовий вхідний рядок, розбиваючи його на вісім блоків по вісім бітів у кожному [14]

$$\omega = (\omega_7, \omega_6, \omega_5, \omega_4, \omega_3, \omega_2, \omega_1, \omega_0). \quad (2.19)$$

До кожного з цих блоків застосовується чотири табличних перестановки $\pi_0, \pi_1, \pi_2, \pi_3$

$$r_j = \pi_j mod 4[\omega_j]. \quad (2.20)$$

Потім отриманий вектор $r = (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$ множиться на матрицю T, елементи якої належать полю $GF(2^8)$.

Далі отриманий вектор $r = (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$ множиться на матрицю T, елементи якої належать полю $GF(2^8)$.

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{pmatrix} = \begin{pmatrix} 01 & 01 & 05 & 01 & 08 & 06 & 07 & 04 \\ 04 & 01 & 01 & 05 & 01 & 08 & 06 & 07 \\ 07 & 04 & 01 & 01 & 05 & 01 & 08 & 06 \\ 06 & 07 & 04 & 01 & 01 & 05 & 01 & 08 \\ 08 & 06 & 07 & 04 & 01 & 01 & 05 & 01 \\ 01 & 08 & 06 & 07 & 04 & 01 & 01 & 05 \\ 05 & 01 & 08 & 06 & 07 & 04 & 01 & 01 \\ 01 & 05 & 01 & 08 & 06 & 07 & 04 & 01 \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{pmatrix}.$$

Для економії часу таблиці підстановок та матрицю T можна попередньо обчислити один раз і використовувати протягом роботи шифру.

Підстановка породжує 64-бітовий вектор q, який розглядається як 64-бітове слово.

Розглянемо рівні захисту генераторів ключових потоків «Струмок-256» та «Струмок-512» (табл. 2.17) [28].

Таблиця 2.17 – Установлені рівні захисту

Генераторів ключових потоків, установлений рівень захисту	Довжина ключа	Обчислювальна складність для найкращої відомої атаки	Обчислювальна складність квантового криптографічного аналізу
«Струмок-256», високий	256	2^{256}	2^{128}
«Струмок-512», надвисокий	512	2^{512}	2^{256}

Отже, «Струмок-256» пропонує хороший баланс між стійкістю до атак та швидкістю роботи, що робить його універсальним генератором ключових потоків для широкого спектра застосувань. Так само «Струмок-512» забезпечує максимальний рівень стійкості до атак, але має нижчу швидкість роботи.

2.5 Практичні аспекти розділу 2

2.5.1 Реалізація алгоритму симетричного шифрування DES

Для кращого розуміння роботи криптоалгоритму DES зашифруємо відкрите повідомлення в програмі CrypTool 2 за допомогою модуля «DES Visualization» [10].

Відповідні поля заповнюємо вхідними даними в форматі HEX:

Plaintext: 78 B2 C7 F8 75 E9 01 D5;

Key: 78 3A D6 0D 26 AE 72 D8 (рис. 2.24).

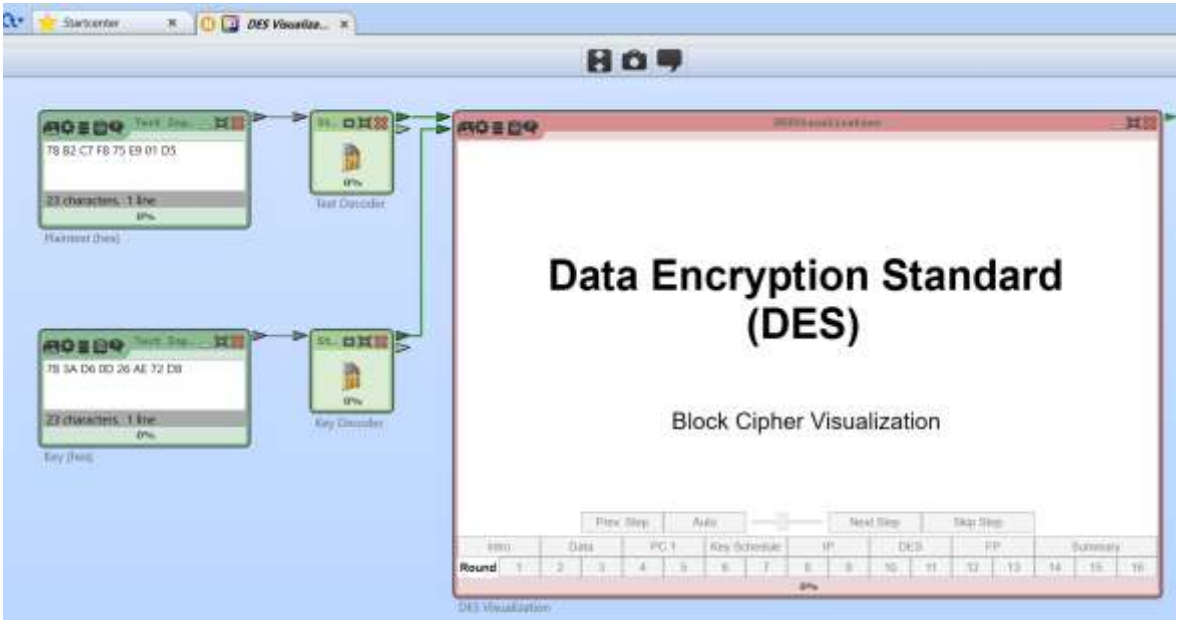


Рисунок 2.24 – Вхідні дані в форматі HEX

Наступним кроком буде переведення вхідних даних в двійковий формат у вигляді 64-бітових блоків, при цьому кожний 8-ий біт ключа видаляється і використовується для контроля парності:

Plaintext: 01111000 10110010 11000111 11111000 01110101 11101001 00000001 11010101;

Key: 01111000 00111010 11010110 00001101 00100110 10101110 01110010 11011000 (рис.2.25).

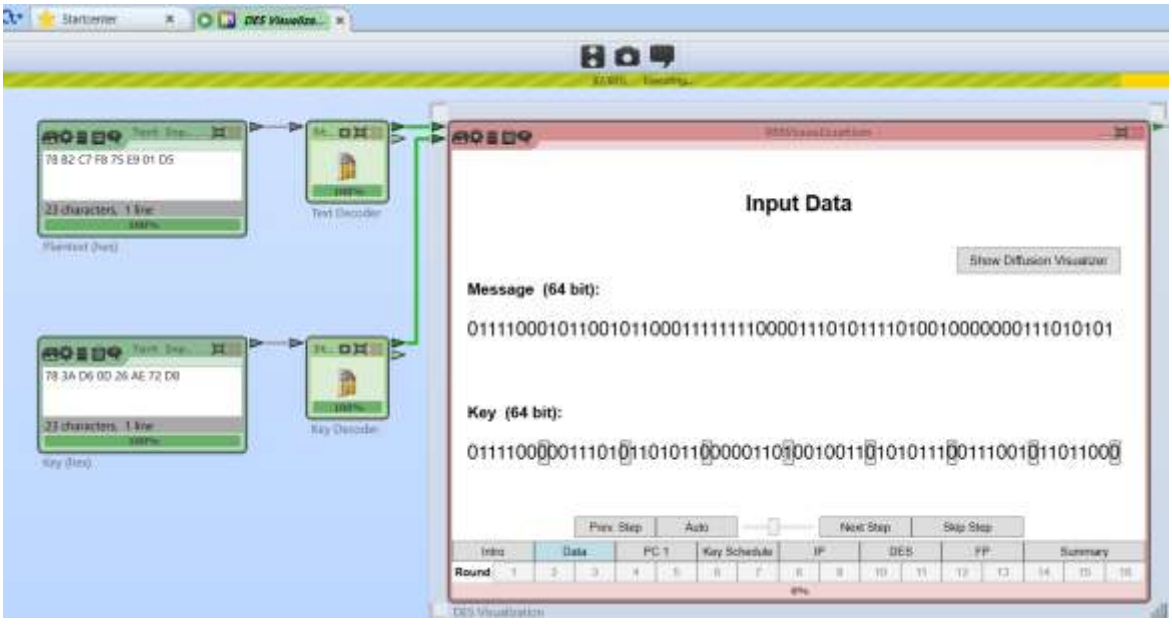


Рисунок 2.25 – Відкритий текст і ключ у двійковому форматі

Наступним кроком є перестановка PC-1 для 56-бітового ключа згідно з таблицею 2.2 (рис. 2.26).

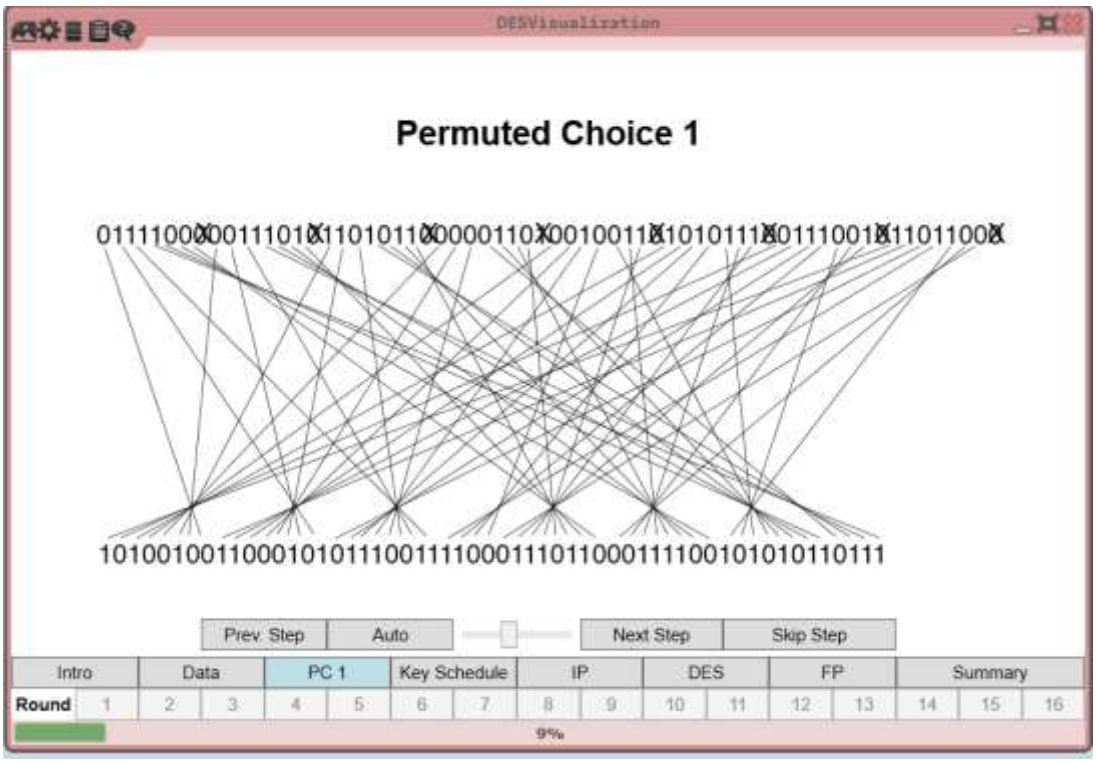


Рисунок 2.26 – Перестановка PC-1

Перестановочний 56-бітовий ключ розділяється на два півблоки по 28 біт, C_0 та D_0. Визначивши C_0 та D_0 створюються шістнадцять блоків C_n та D_n, кожен з яких формується з попередньої пари C_{n-1} та D_{n-1} та циклічного зсуву згідно з таблицею 2.3, залежно від раунду (рис. 2.27).

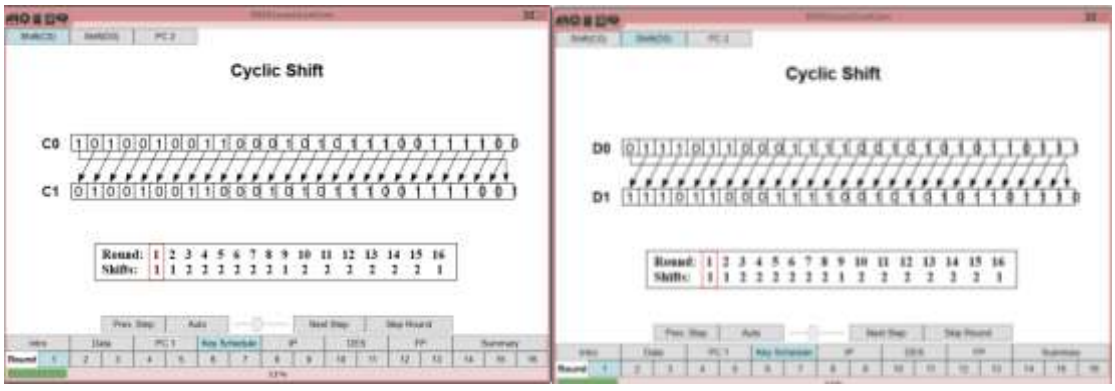


Рисунок 2.27 – Циклічні зсуви C_{n-1} та D_{n-1}

Після циклічного зсуву півблоки C_0 та D_0 об’єднуються в 56-бітову послідовність та застосовується перестановка PC-2 (рис. 2.28) з ущільненням до 48 біт згідно з табл. 2.4.

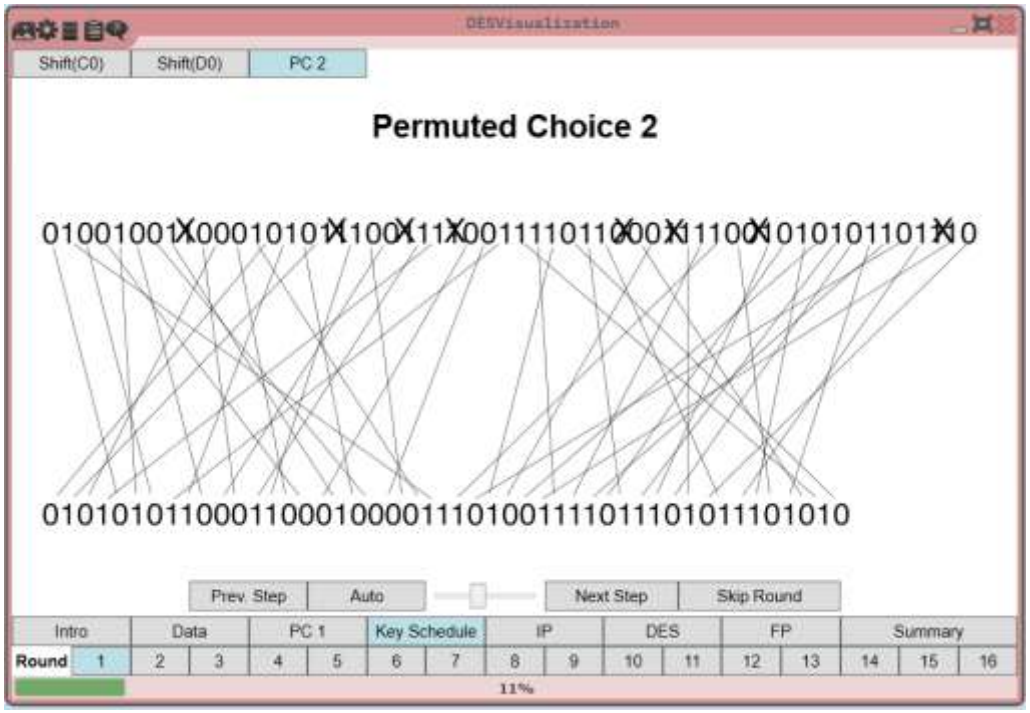


Рисунок 2.28 – Перестановка PC-2

Отже, результатом першого циклу є отриманий раундовий ключ K_1 (рис. 2.29).

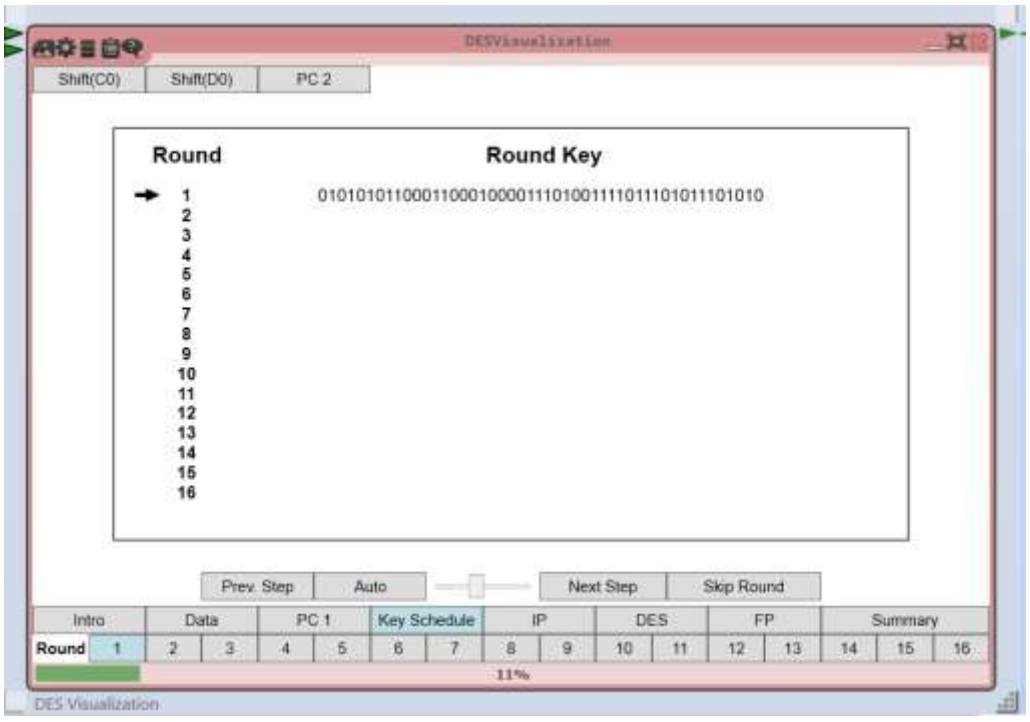


Рисунок 2.29 – Раундовий ключ K_1

Застосувавши таблиці перестановок до кожної з пар півблоків C_n та D_n будуть сформовані всі шістнадцять раундових ключів K_n (рис. 2.30).

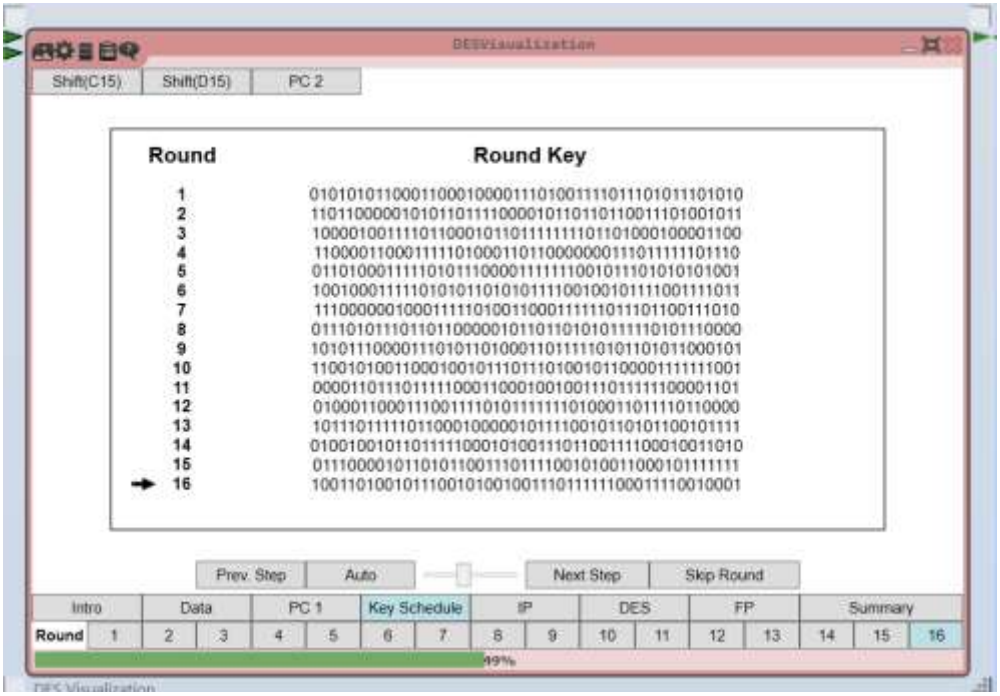


Рисунок 2.30 – Раундові ключі K_n

Після того як отримано раундові ключі, шифрування відкритого тексту відбувається протягом 16 раундів. Застосуємо до відкритого тексту Plaintext початкову перестановку IP згідно з таблицею 2.1 (рис. 2.31):

Plaintext: 01111000 10110010 11000111 11111000 01110101 11101001 00000001 11010101;

IP: 10111101 10011011 10010100 11110100 10101110 00111011 00101001 00000110.

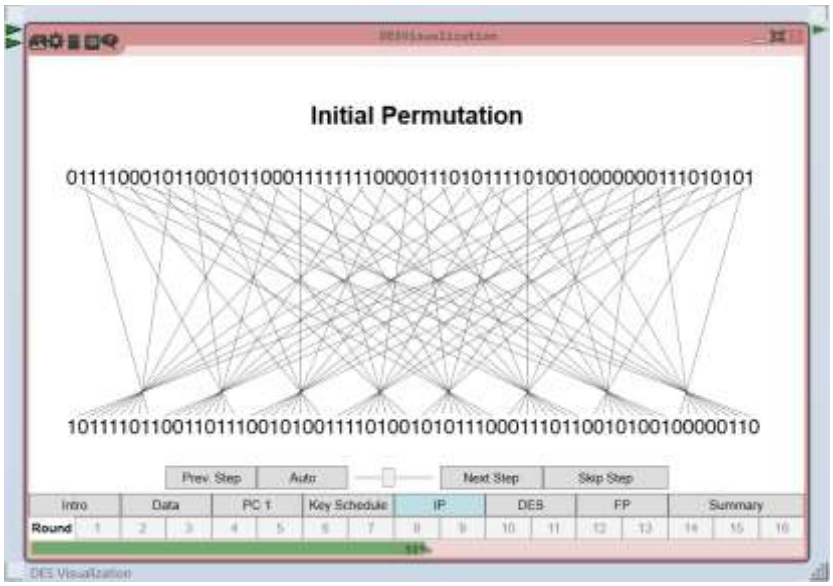


Рисунок 2.31– Початкова перестановка IP

Тобто 58-й біт Plaintext «1» стає першим бітом перестановки IP, 50-й біт Plaintext «0» стає другим бітом IP тощо.

Після початкової перестановки IP вхідні дані розбиваються на два підблоки по 32 біти, лівий L_n та правий R_n:

L_0: 10111101 10011011 10010100 11110100;
R_0: 10101110 00111011 00101001 00000110.

Наступним кроком будуть 16 ітерацій за допомогою функції *f*. Аргументами для функції f є 48-бітовий ключ k_i та 32-бітовий півблок, що розширюється до 48 біт за допомогою таблиці розширення (рис. 2.32).

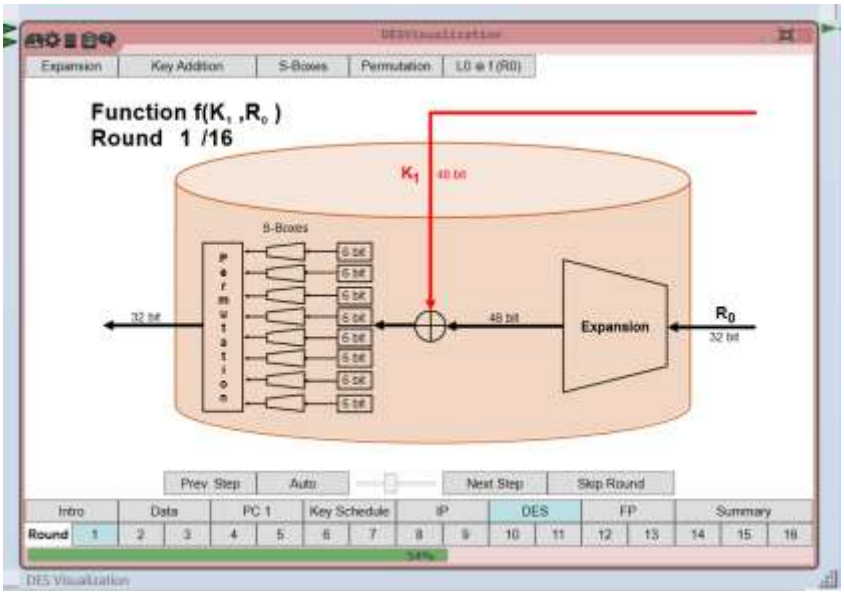


Рисунок 2.32 – Функція $f(k_i, R_{i-1})$

Розширення півблока R_i (Expansion) з використанням перестановки розширення E зображено на рисунку 2.33.

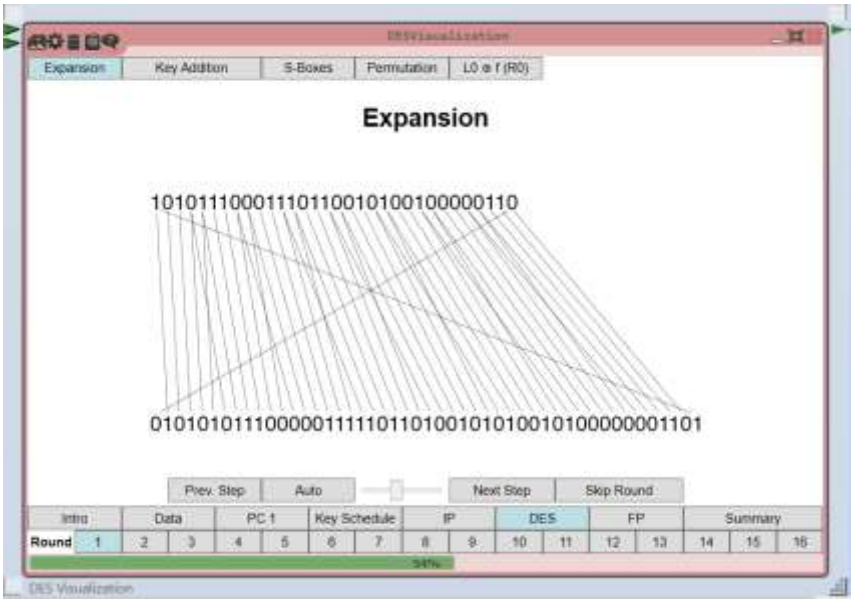


Рисунок 2.33 – Розширення півблока R_i

Наступним кроком буде порозрядне додавання за модулем 2 правого півблока з 48-бітовим раундовим ключем (рис. 2.34).

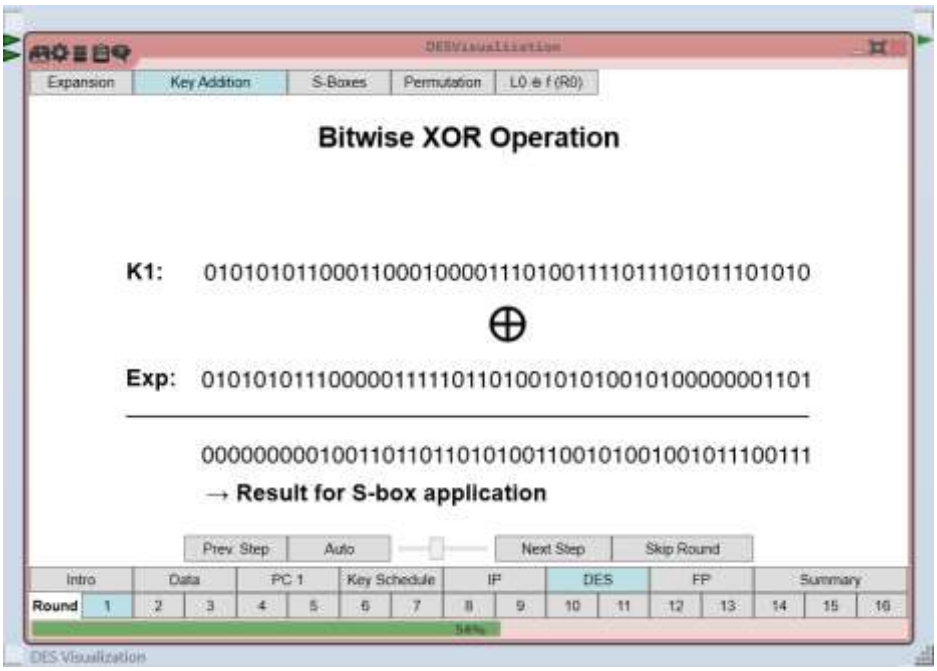


Рисунок 2.34 – Порозрядне додавання за модулем 2

Результат попередньої операції проходить через *S*-блоки, виконуючи нелінійну заміну (рис. 2.35).

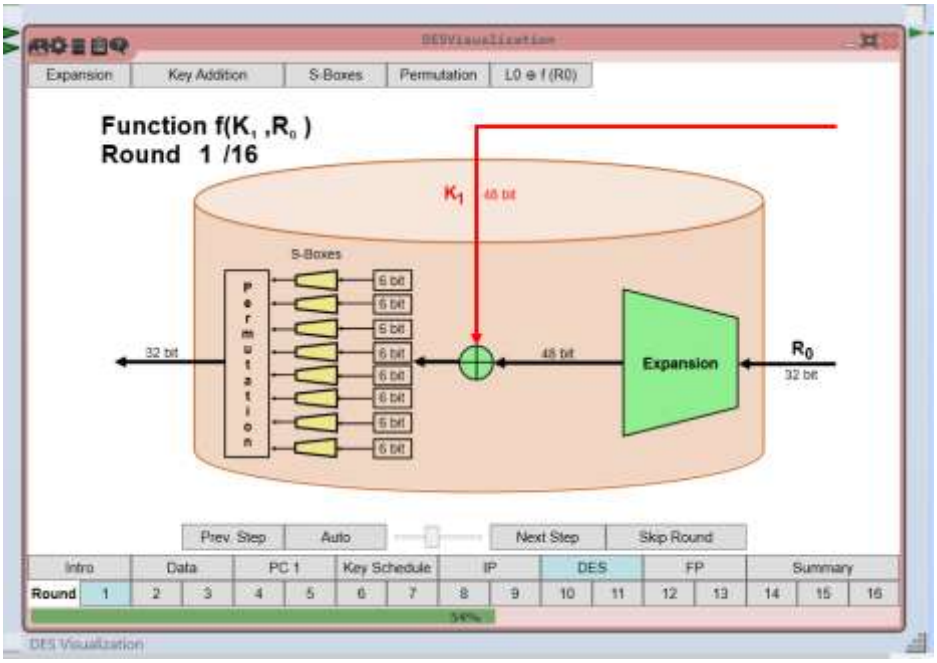


Рисунок 2.35 – Нелінійна заміна

Кожні 6 бітів 48-бітової послідовності замінюються на 4 біти (1-й та 6-й біти відкидаються) за допомогою таблиць заміни (рис. 2.36).

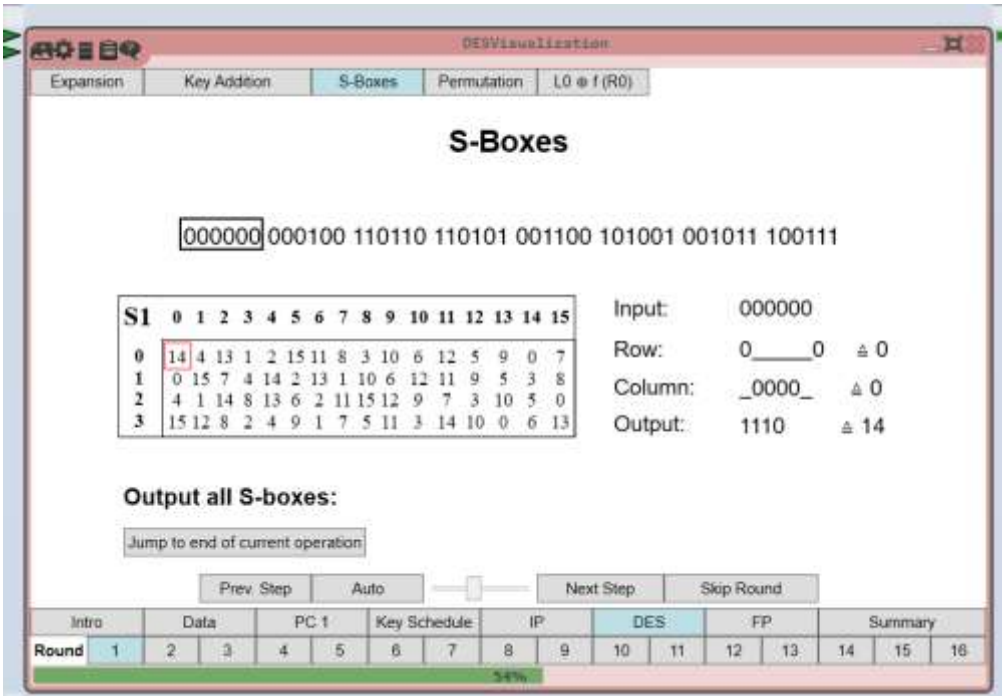


Рисунок 2.36 – Перетворення S-Boxes

Після проходження всіх *S*-блоків отримаємо вихідну бітову послідовність: 11101000110001011011100110010111 (рис. 2.37).

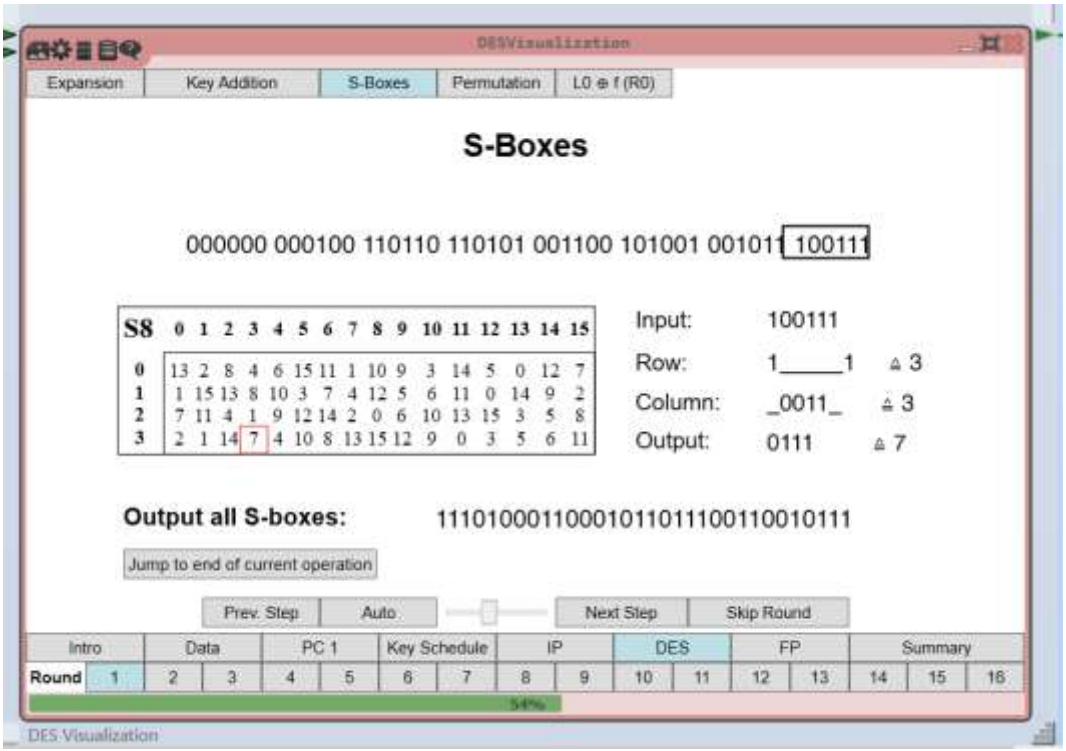


Рисунок 2.37 – Вихідна бітова послідовність після проходження *S*-блоків

Згідно з таблицею 2.5 (Перестановка за допомогою P-блоків) отримаємо правий півблок R_j нової пари півблоків (рис. 2.38).

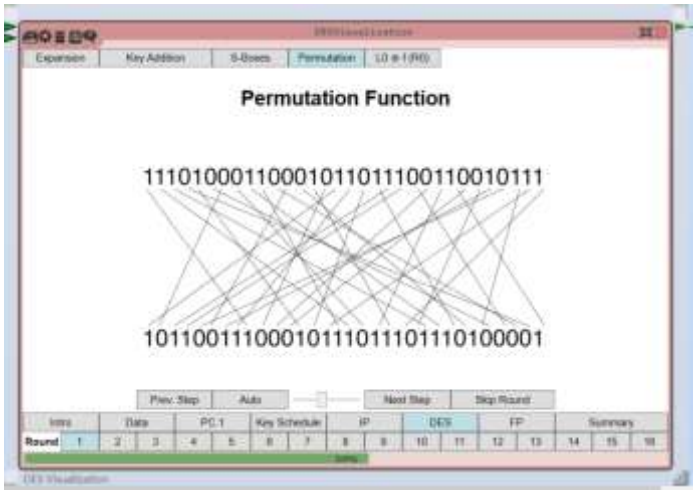


Рисунок 2.38 – Перестановка Permution Function

Результат Permution Function за допомогою операції XOR об’єднується з L_i (рис. 2.39) і після чого L_i та R_i міняються місцями (рис. 2.40).

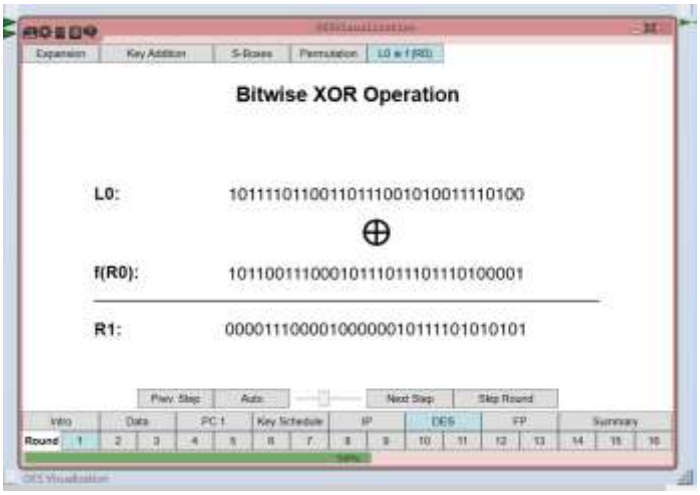


Рисунок 2.39 – Виконання операції XOR

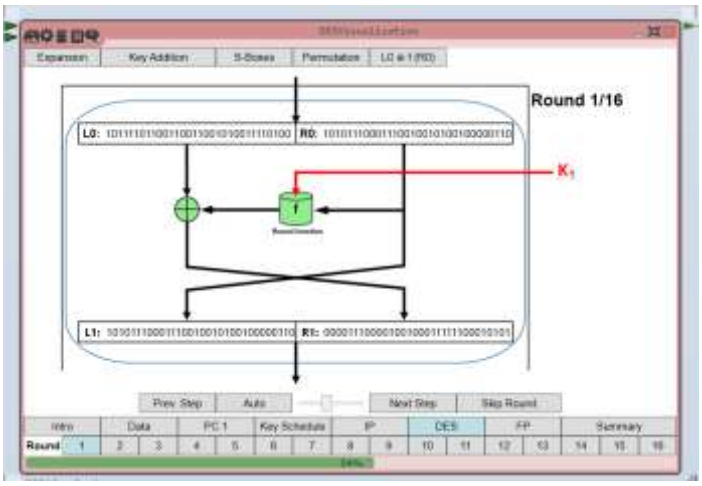


Рисунок 2.40 – Перестановка L_i та R_i

Результатом $Round_{16}$ є отримана послідовність:

L_{16} : 11101100001010111000100011011000
R_{16} : 01100110111011110100110110010001 (рис. 2.41).

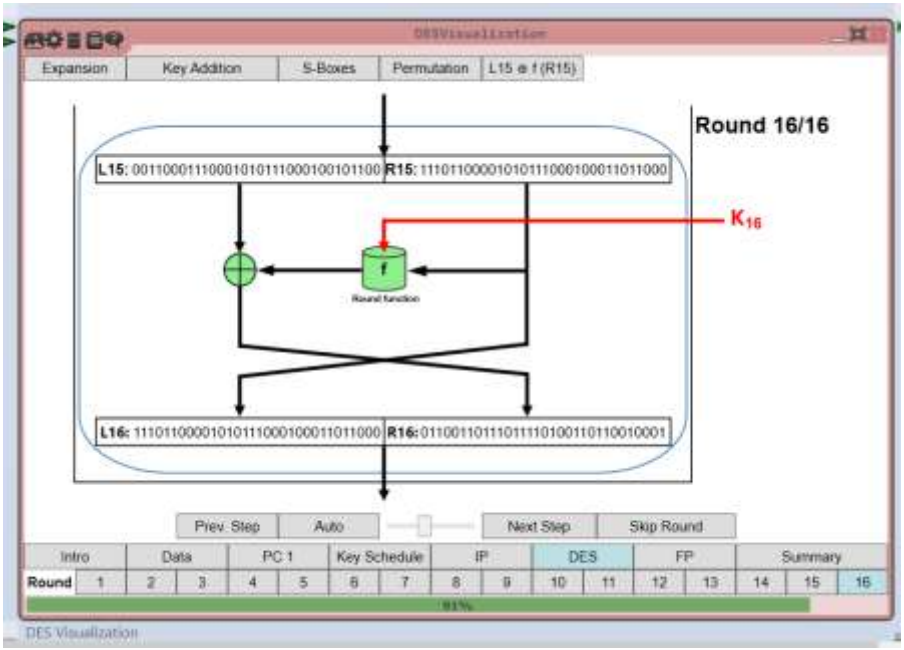


Рисунок 2.41 – Результат виконання $Round_{16}$

Після чого міняємо місцями L_{16} та R_{16}, отримаємо послідовність R_{16} L_{16}, та виконуємо фінальну перестановку IP^{-1} відповідно до таблиці 2.6 (рис. 2.42).

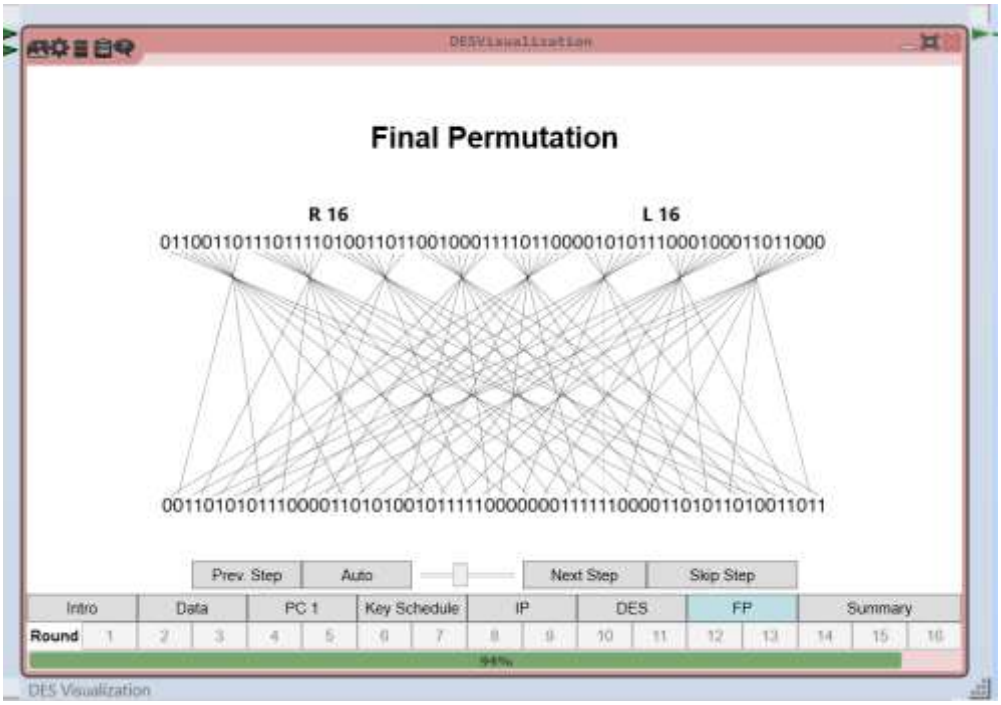


Рисунок 2.42 – Фінальна перестановка IP^{-1} послідовності $R_{16}L_{16}$

В результаті фінальної перестановки IP^{-1} отримано послідовність (рис. 2.43):

00110101011100001101010010111110000000111111000011010011101

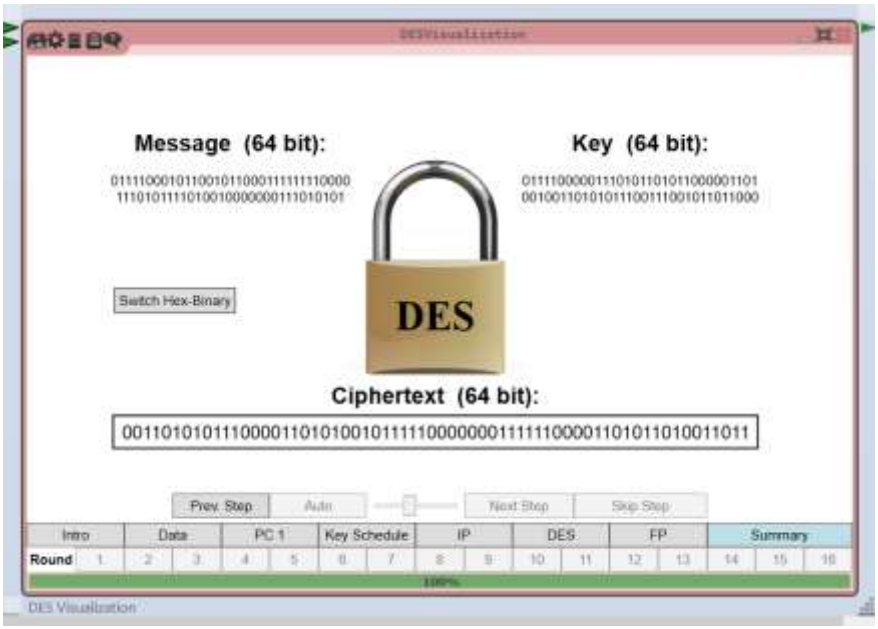


Рисунок 2.43 – Зашифроване повідомлення

Повідомлення Plaintext: 78 B2 C7 F8 75 E9 01 D5 було зашифровано криптоалгоритмом DES за допомогою ключа Key: 78 3A D6 0D 26 AE 72 D8 та отримано Ciphertext: A3 08 4F 19 02 38 5E EA (рис. 2.44).

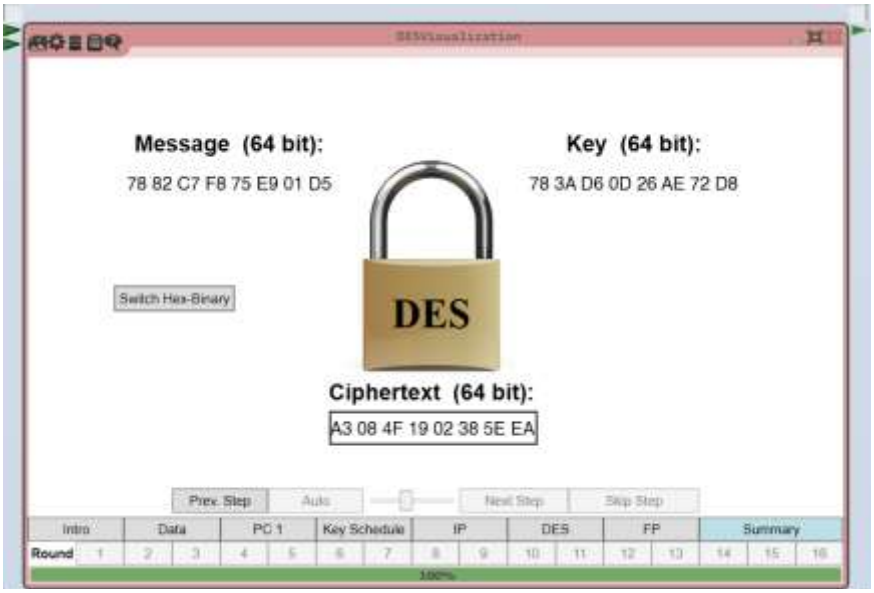


Рисунок 2.44 – Вихідні дані в форматі HEX

Розшифрування даних виконується у зворотному порядку, дотримуючись тих самих кроків, що й були описані вище, але змінюючи порядок застосвання підключів.

2.5.2 Застосування криптоалгоритму AES для шифрування повідомлень

Розглянемо процес шифрування криптоалгоритмом AES.

Зашифруємо відкрите повідомлення в програмі CrypTool 2 за допомогою модуля «AES Visualization» [10].

Відповідні поля заповнюємо вхідними даними в форматі HEX:

PlainText: 4A41F6B85A8DF7828C8D98D6F8E9045D

Key: 1CDE1816F5A9D2A6A4F71D2809CF4F5A (рис. 2.45).

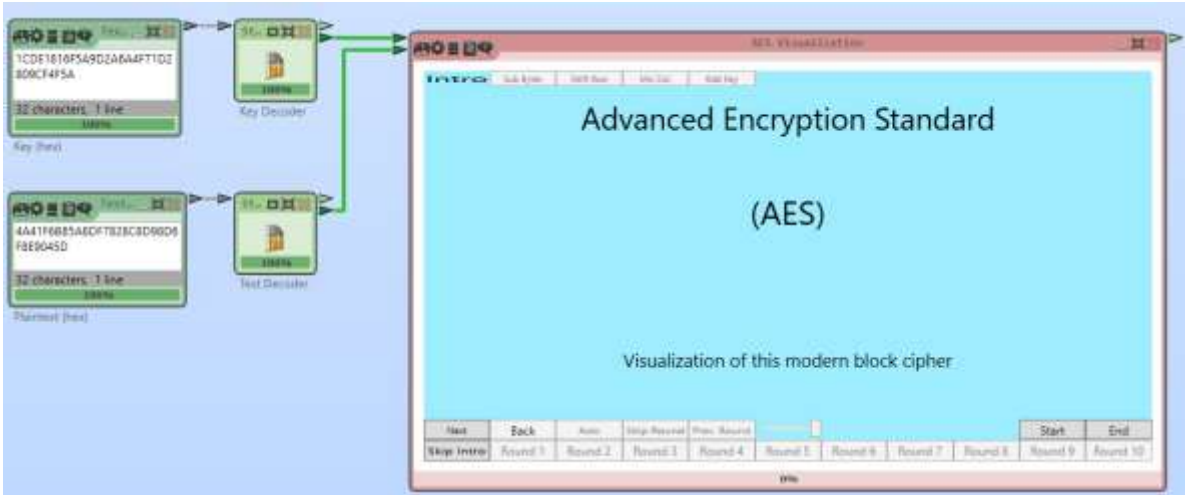


Рисунок 2.45 – Вхідні дані в форматі HEX

Для криптоалгоритму AES-128 ключ із 128 бітів, поділених на 16 байтів, задається матрицею станів 4 × 4 (*States*) [19], послідовність розміщується по стовпцях вниз (рис. 2.46).

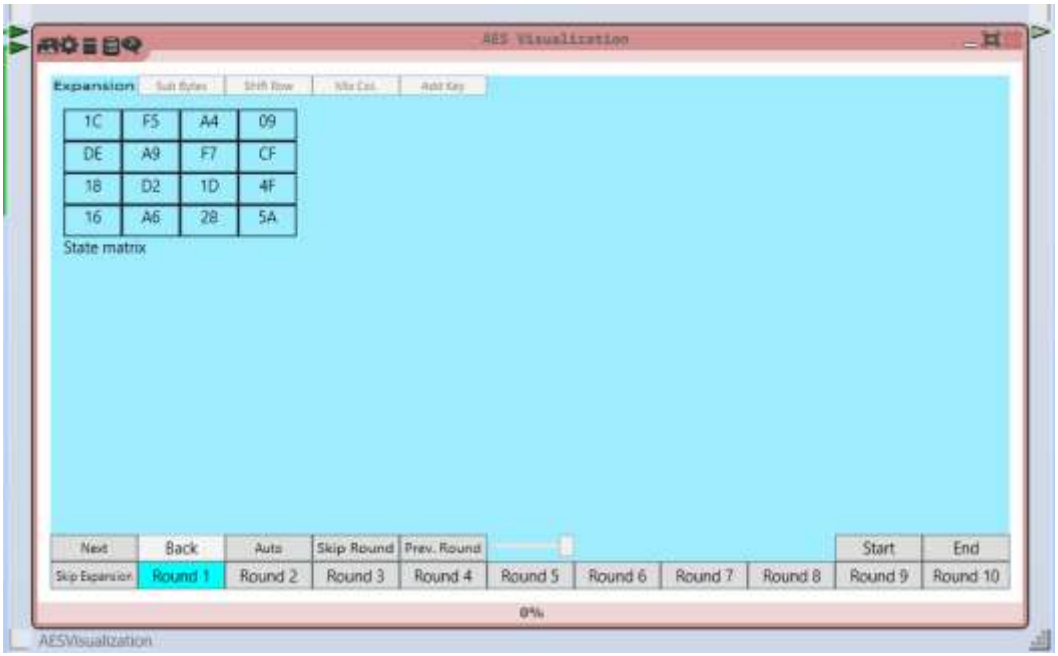


Рисунок 2.46 – Матриця станів

Отримаємо ключ шифру з чотирьох слів:

$w_0 = 1C\ DE\ 18\ 16,$
$w_1 = F5\ A9\ D2\ A6,$
$w_2 = A4\ F7\ 1D\ 28,$
$w_3 = 09\ CF\ 4F\ 5A.$

AES-128 має початковий ключ Key, що використовується для отримання 11 раундових ключів, кожен із яких становить чотири слова w_i, до яких застосовується алгоритм *Expand Key*. В підсумку буде отримано $4(N_r + 1)$ 32-бітових слів. До слова w_3 застосовуємо перетворення *RotWord*, однобайтовий циклічний зсув вліво та отримаємо $w'_3 = CF\ 4F\ 5A\ 09$ (рис. 2.47).

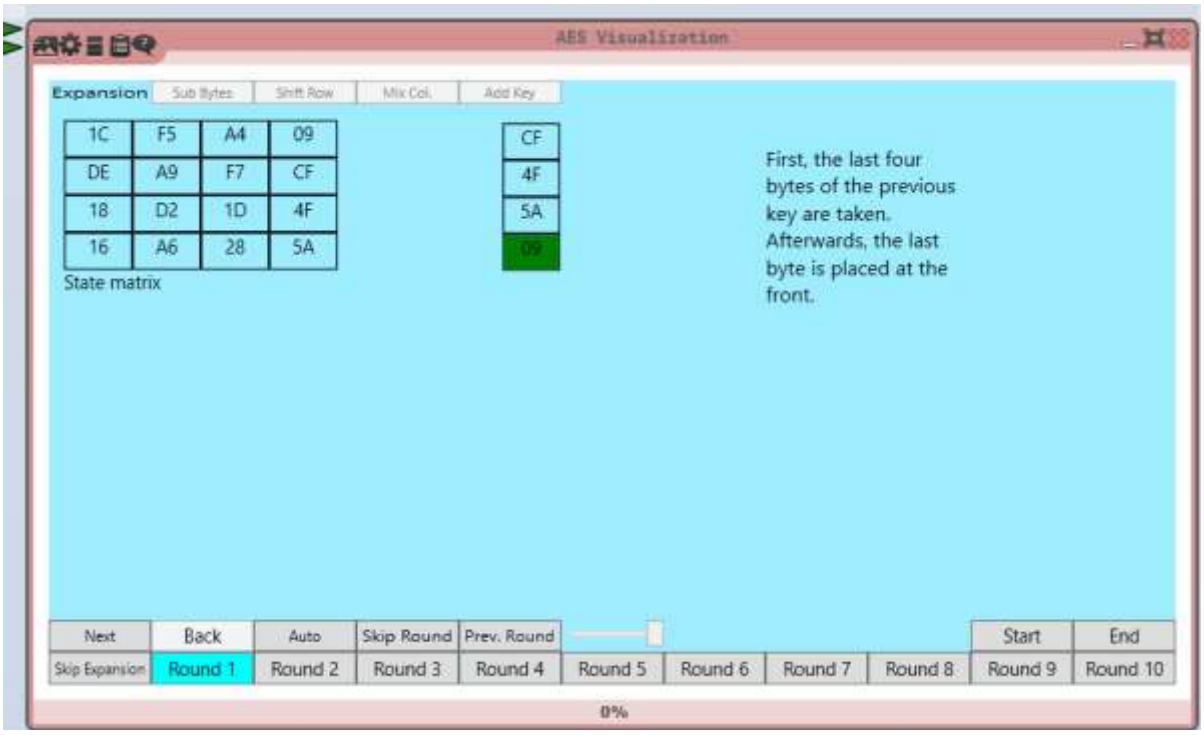


Рисунок 2.47 – Перетворення *RotWord*

До кожного з чотирьох байтів зміненого слова w'_3 застосовуємо операцію заміни байтів *SubBytes* згідно з таблицею 1.1 (рис. 2.48–2.49).

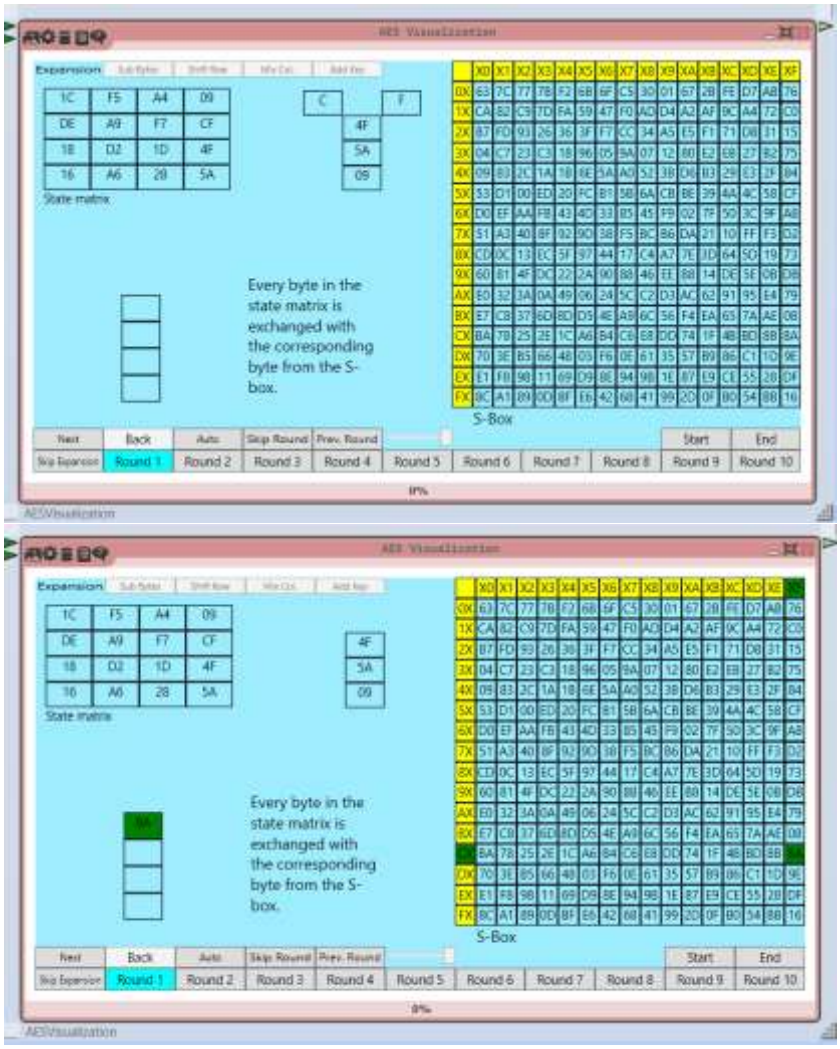


Рисунок 2.48 – Процес перетворення *SubBytes*

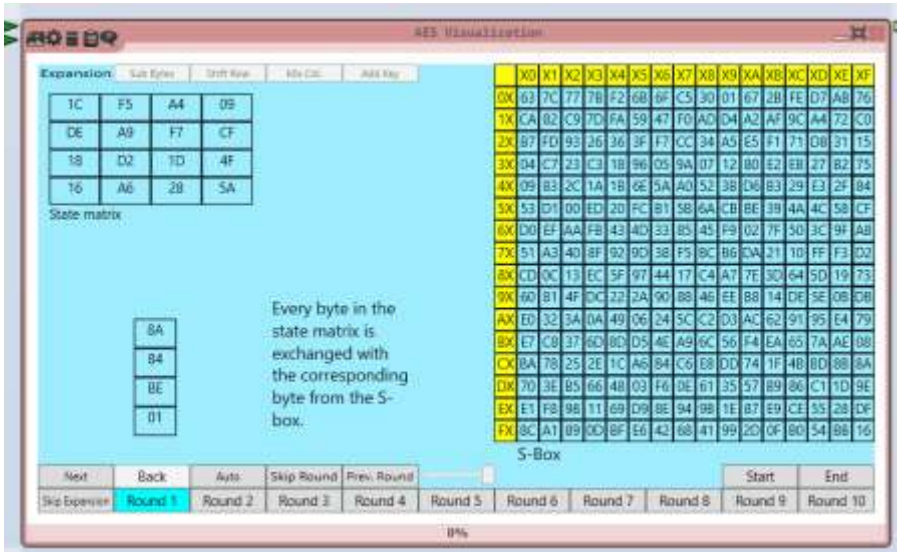


Рисунок 2.49 – Отриманий результат перетворення *SubBytes*

Наступним кроком є додавання за модулем 2, операція *XOR*, байтів стовпця, отриманих на попередньому кроці, та раундової константи R_{con}

(див. табл. 2.2). В результаті додавання раундової константи $(01, 00, 00, 00)$ отримаємо: $g(w_3) = (88, 84, BE, 01)$ (рис. 2.50).

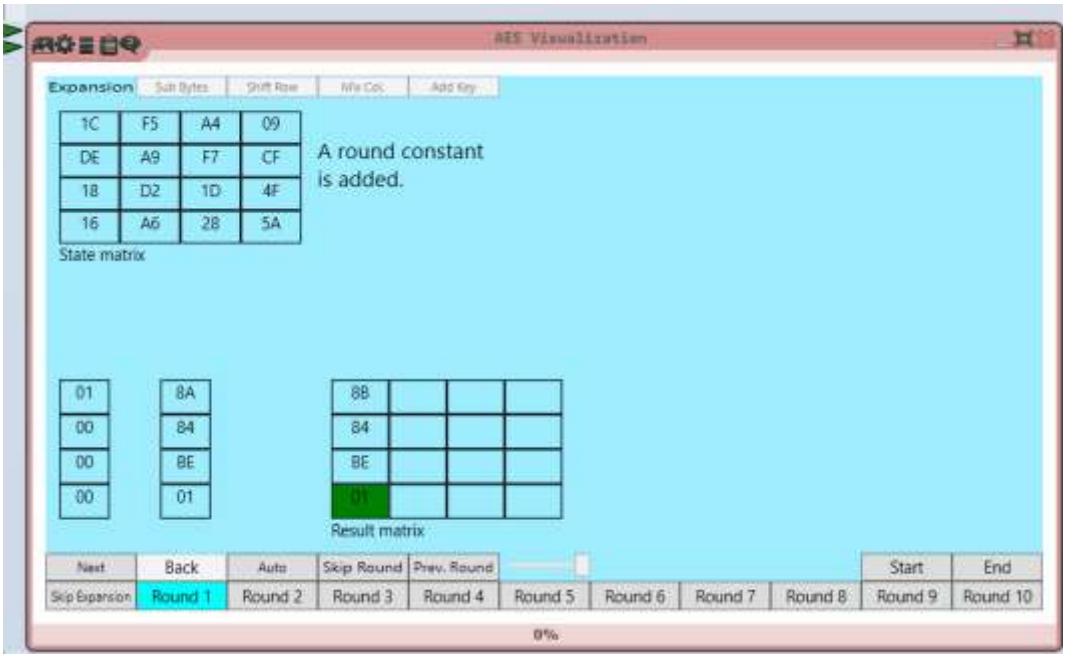


Рисунок 2.50 – Перетворення R_{con}

Застосувавши формулу $w_4 = w_0 \oplus g(w_3) = (97{,}5A, A6, 17)$, отримаємо w_4 (рис. 2.51).

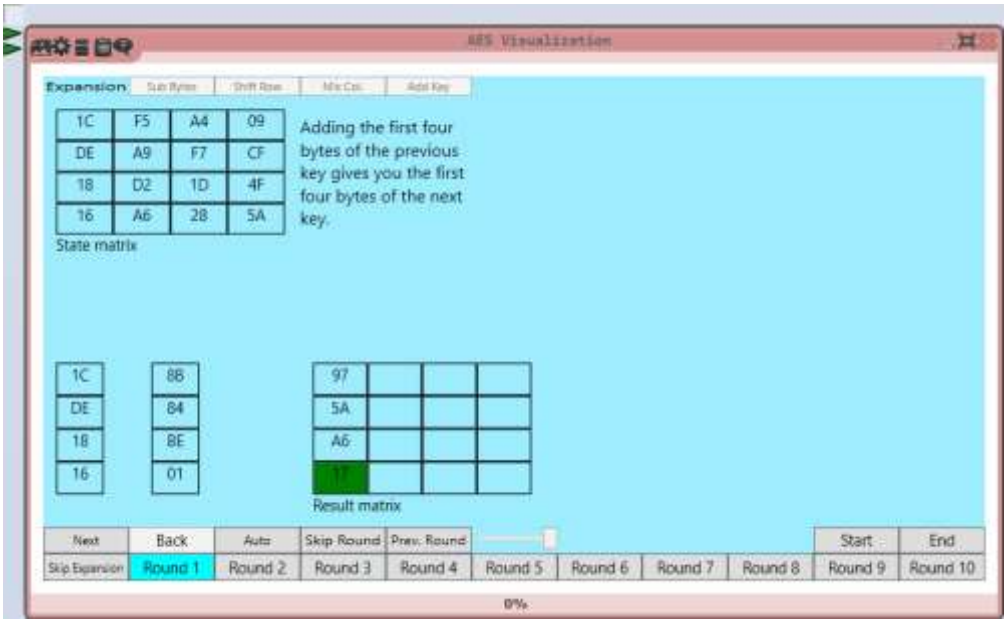


Рисунок 2.51 – Отримане нове слово w_4

Для знаходження кожного слова w_i нового раундового ключа виконуємо операцію *XOR* між байтами слова w_i нової матриці стану та байтами w_{i+1} попереднього ключа (рис. 2.52).

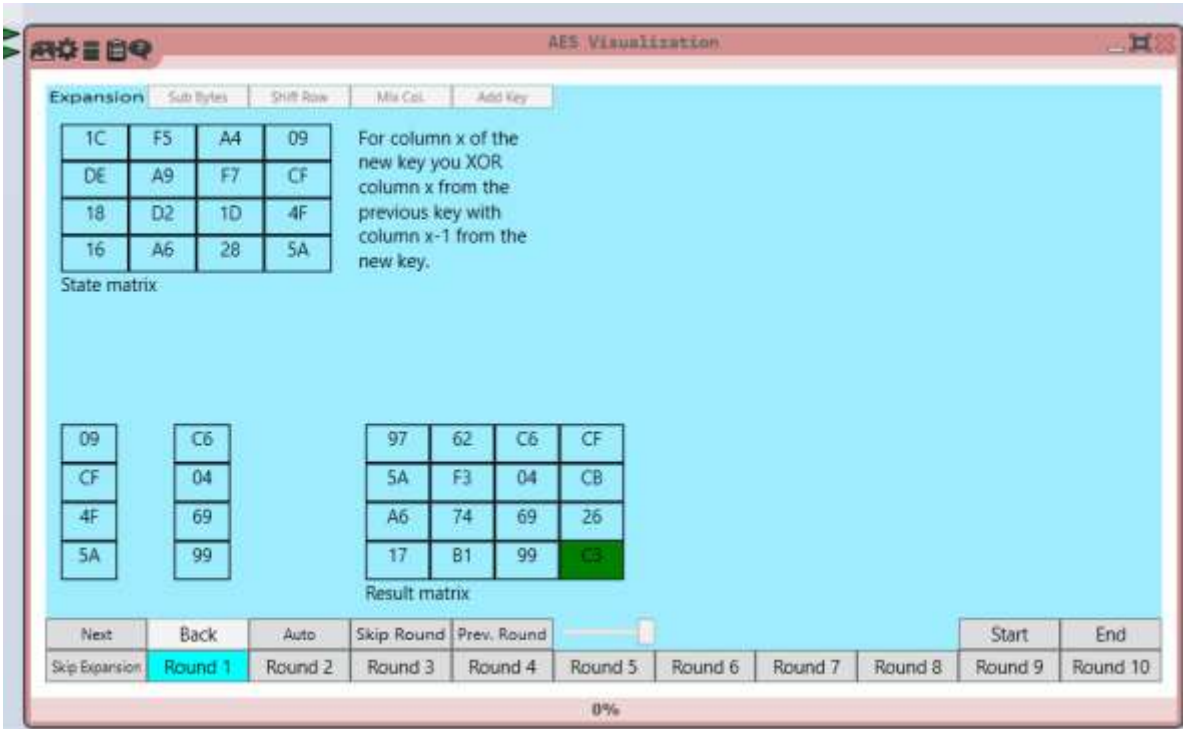


Рисунок 2.52 – Операція *XOR*

Отримана так послідовність слів w_4, w_5, w_6, w_7 є ключем для 1-го раунду та матрицею станів для наступного раундового ключа (рис. 2.53).

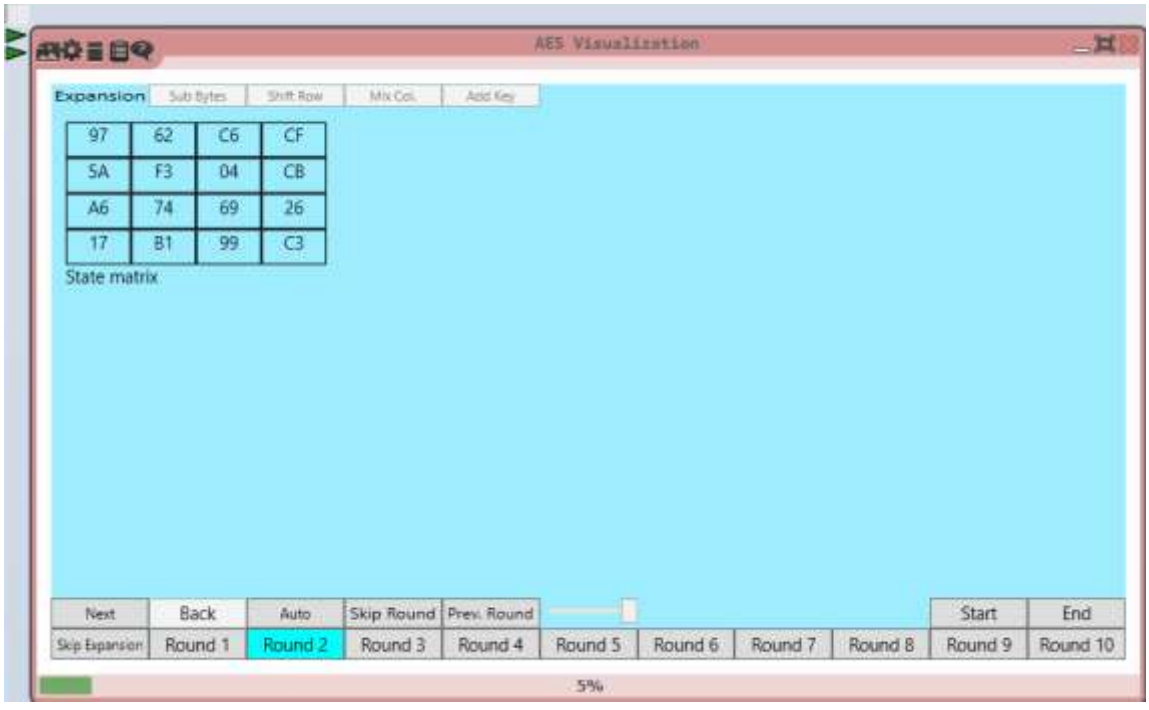


Рисунок 2.53 – Ключ 1-го раунду Key_1

Після виконання 10 раундів маємо такі раундові ключі:
Key_0: 1CDE1816F5A9D2A6A4F71D2809CF4F5A;
Key_1: 975AA61762F374B1C6046999CFCB26C3;

Key_2: 8AAD889DE85EFC2C2E5A95B5E191B375;
Key_3: 0FC0B065E79E4C49C9C4D9FC28556A8A;
Key_4: FBC2CE511C5C8218D5985BE4FDCD316E;
Key_5: 560551054A59D31D9FC188F9620CB997;
Key_6: 8853D9AFC20A0AB25DCB824B3FC73BDC;
Key_7: 0EB15FDACCBB55689170D723AEB7ECFF;
Key_8: 277F493EEBC41C567AB4CB75D403278A;
Key_9: 47B33776AC772820D6C3E05502C0C7DF;
Key_{10}: CB75A90167028221B1C16274B301A5AB.

Отримані раундові ключі використовують в наступних десяти раундах шифрування.

Перед першим раундом виконується перетворення AddRoundKey: операція *XOR* матриці стану State відкритого тексту та матриці початкового ключа шифру Key_0. Перетворення, виконані в одному раунді, позначають Result matrix (State-Matrix, Key-Matrix), де змінна State-Matrix – матриця, що описує дані на вході раунду та на його виході після шифрування; змінна Key-Matrix – матриця, містить раундовий ключ (рис. 2.54).

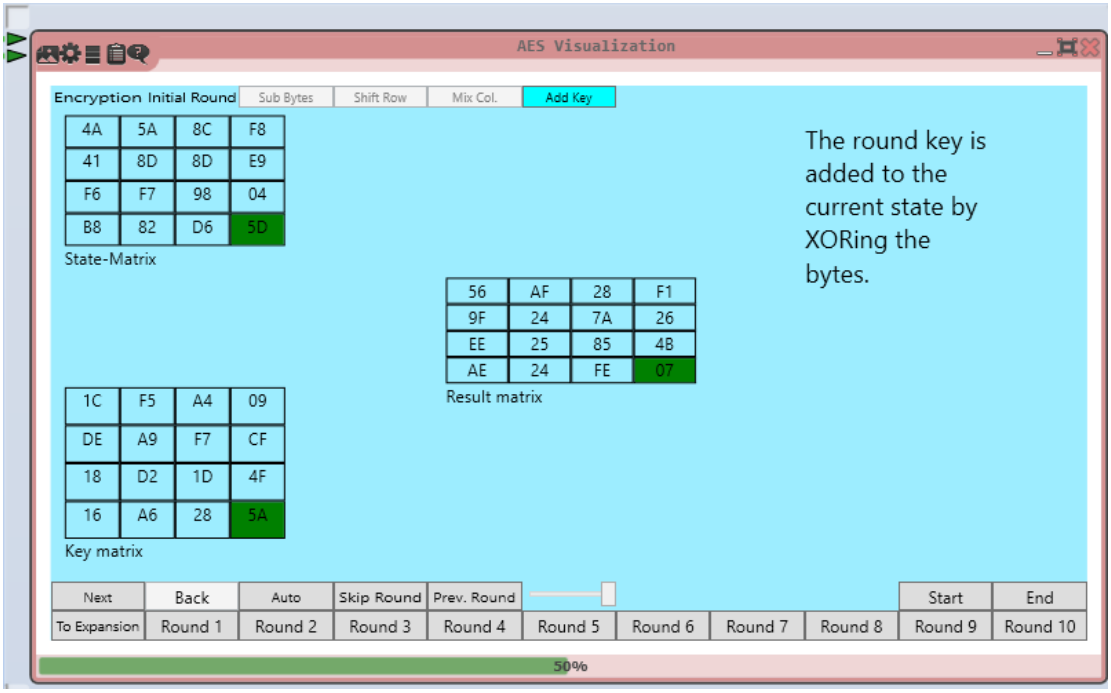


Рисунок 2.54 – Перетворення AddRoundKey

Застосовуємо таблицю підстановки *SubBytes*, згідно з таблицею 1.1, до кожного байта матриці стану. Значення підстановки визначатиметься перетином рядка з індексом '5' і стовпця з індексом '6' у таблиці, так що 56 = {B1}(рис. 2.55).

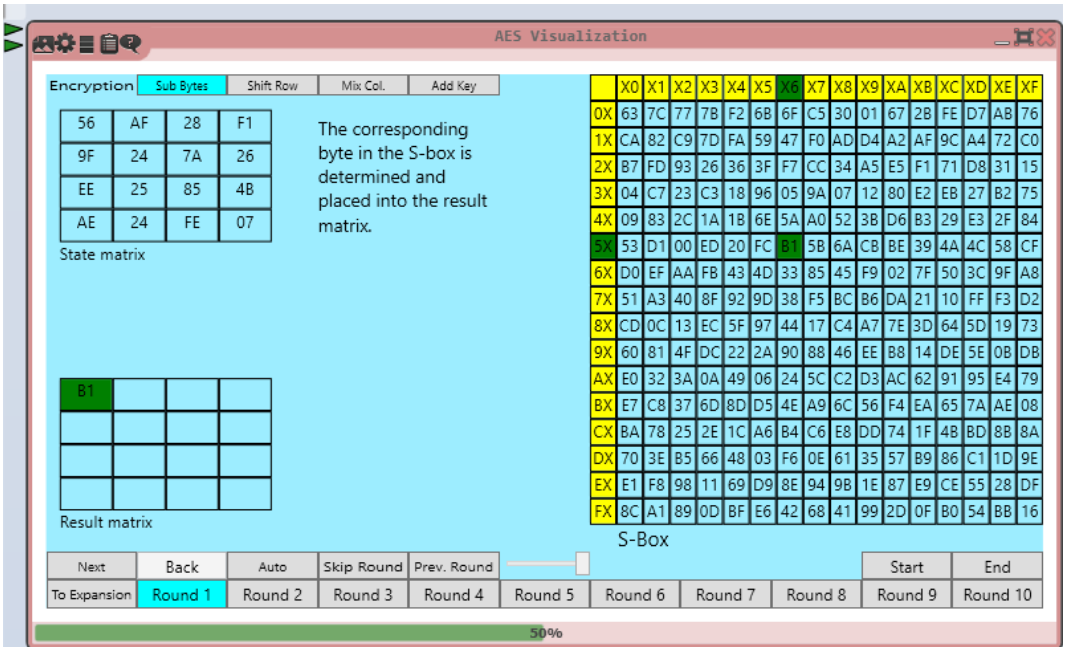


Рисунок 2.55 – Перетворення *SubBytes*

Як результат отримаємо підсумкову матрицю (рис. 2.56).

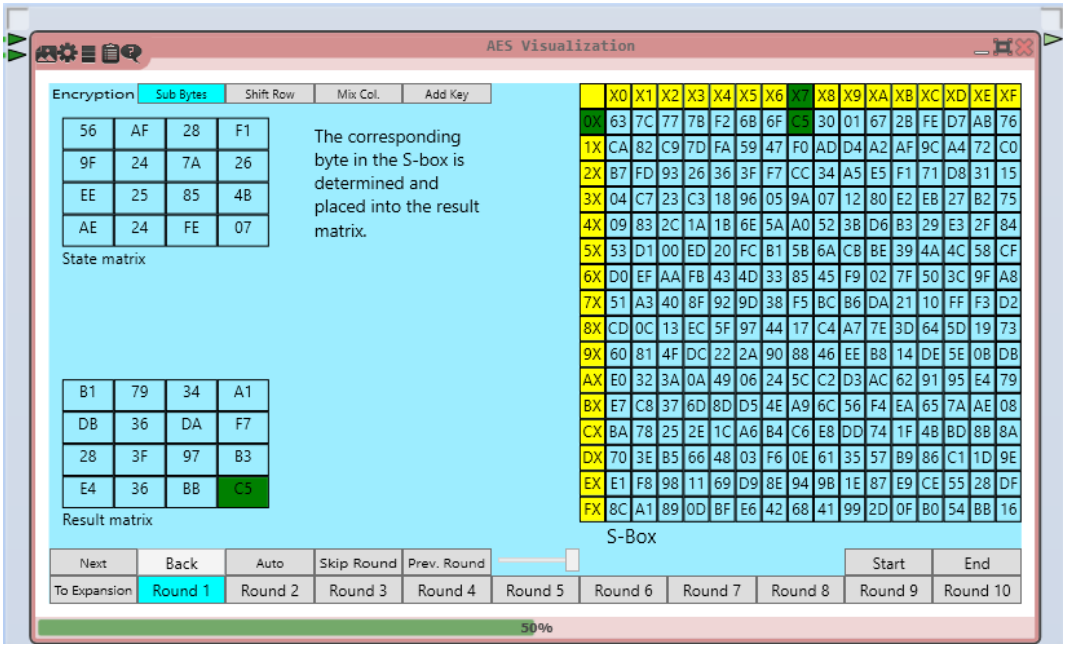


Рисунок 2.56 – Підсумкова матриця

Перетворення *ShiftRows* застосовується до рядків матриці *States*. Це перетворення стану, при якому байти в останніх трьох рядках стану циклічно зсуваються. Байти в першому рядку взагалі не зсуваються. Байти у другому рядку зсунуті на одну позицію, у третьому рядку – на дві позиції, а байти в четвертому рядку зсунуті на три позиції вліво. Крайні ліві байти в кожному рядку переміщуються в праву частину того самого рядка (рис. 2.57).

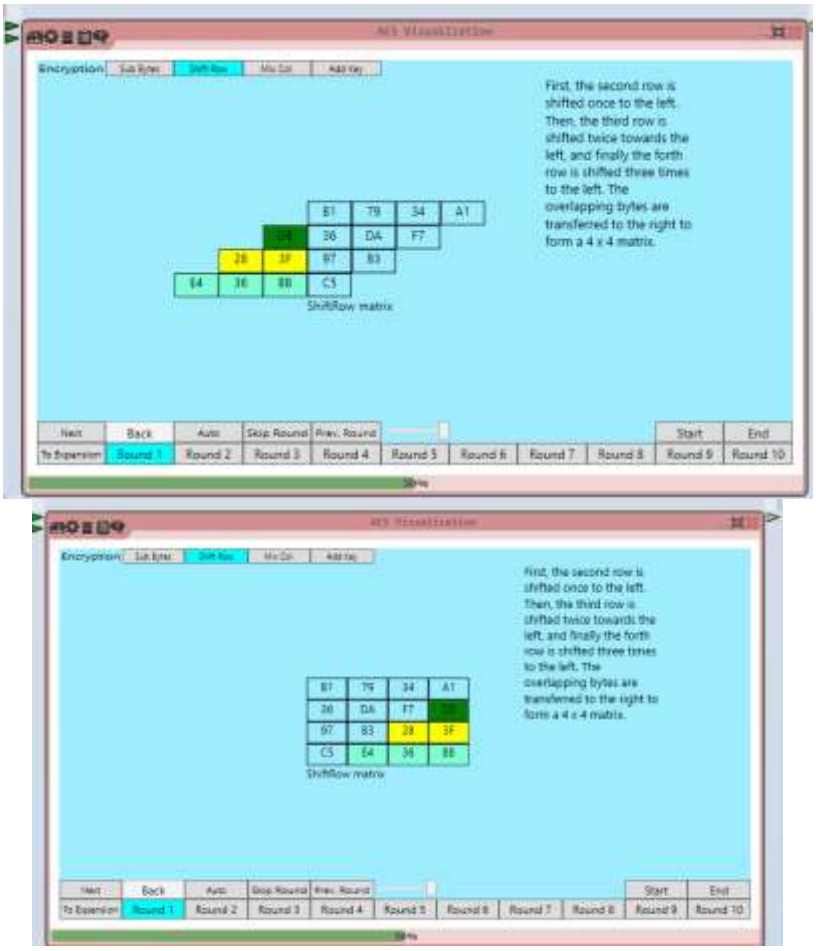


Рисунок 2.57 – Перетворення *ShiftRows*

Перетворення *MixColumns* поєднує кожен стовпець *State matrix* виконанням множення матриці з фіксованою *Multiplication matrix* (рис. 2.58).

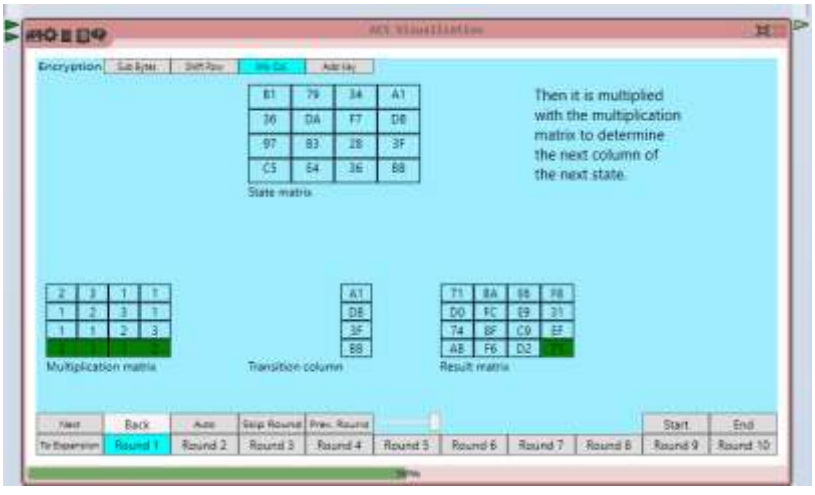


Рисунок 2.58 – Перетворення *MixColumns*

Отримаємо підсумкову матрицю

{71D074ABBAFC8FF6E6E9C9D2F831EF71}.

Завершальне перетворення 1-го раунду *AddRoundKey* є перетворенням стану, у якому раундовий ключ Key_1 поєднується зі State-Matrix із застосуванням побітової операції XOR (рис. 2.59).

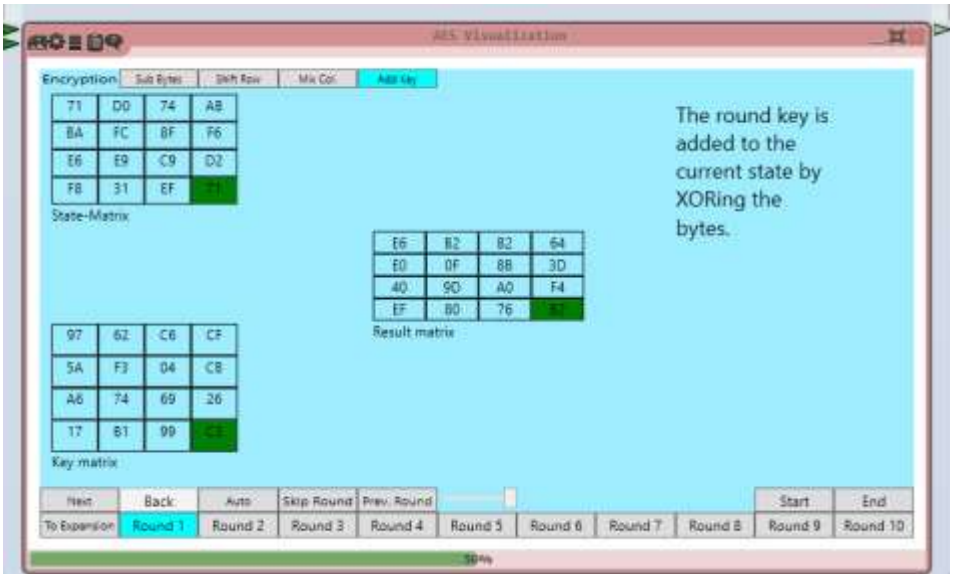


Рисунок 2.59 – Перетворення 1-го раунду *AddRoundKey*

У процесі проходження кожного з 9-ти раундів виконуються перетворення *SubBytes*, *ShiftRows*, *MixColumns* та *AddRoundKey*. У фінальному раунді виконуються ті самі перетворення, крім *MixColumns*. Після проходження 10-ти раундів отримано шифротекст (рис. 2.60).

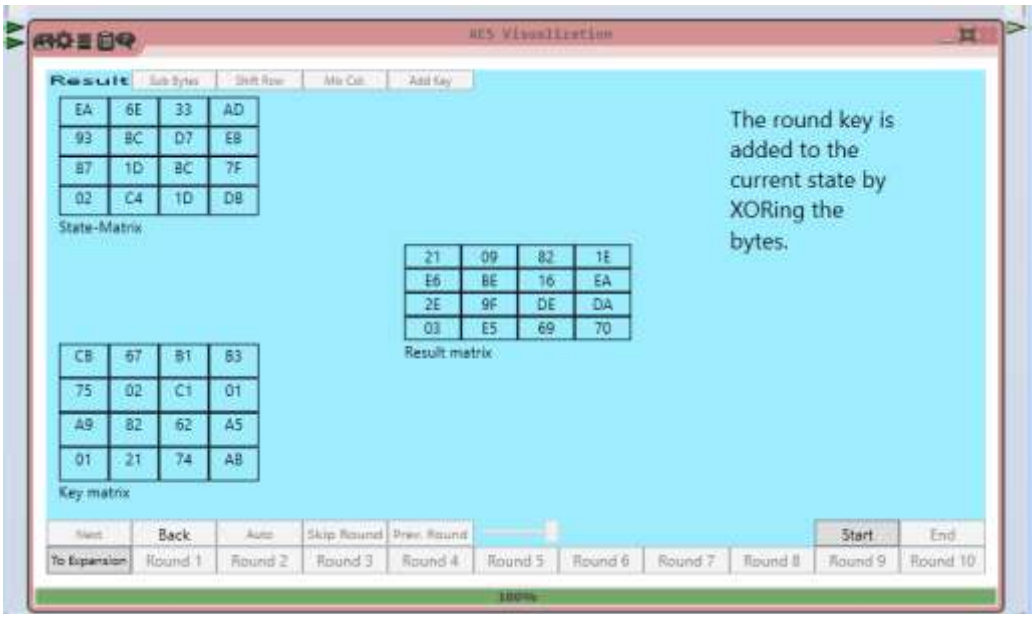


Рисунок 2.60 – Вихідне повідомлення

Отже, криптоалгоритмом AES-128 було зашифровано повідомлення PlainText: 4A41F6B85A8DF7828C8D98D6F8E9045D за допомогою ключа

Key: 1CDE1816F5A9D2A6A4F71D2809CF4F5A та отримано Ciphertext: 21E62E0309BE9FE58216DE691EEADA70.

2.6 Контрольні питання

1. Що є основним елементом шифру DES?
2. Опишіть структуру алгоритму DES.
3. Наскільки криптографічно стійким є шифрування блоковим шифром DES?
4. Які існують режими використання алгоритму шифрування DES?
5. Які етапи охоплює процес шифрування даних за допомогою алгоритму «Струмок»?
6. Опишіть основні функції криптоалгоритму «Струмок».
7. На яких двох ключових компонентах ґрунтується генератор ключових потоків «Струмок»?
8. Порівняйте шифр «Струмок» з шифром SNOW-2.0.
9. Як генеруються раундові ключі в AES?
10. Яка структура блока даних в AES?
11. Як змінюється кількість раундів шифрування залежно від розміру ключа в AES?
12. Які криптографічні властивості роблять AES стійким до атак?
13. Які режими роботи AES використовуються для шифрування повідомлень різної довжини?
14. Як впливає довжина ключа на стійкість криптоалгоритму AES до атак?
15. Порівняйте криптоалгоритм AES з симетричними шифром DES.
16. Опишіть структуру раунду в криптоалгоритмі «Калина».
17. Які відомі атаки на алгоритм «Калина»? Які їхні основні принципи?
18. Як оцінюється стійкість алгоритму «Калина» до різних типів атак?
19. Які основні операції використовуються в алгоритмі «Калина»?
20. Які переваги алгоритму «Калина» порівняно з іншими сучасними блоковими шифрами?

РОЗДІЛ 3 КРИПТОСИСТЕМИ З ВІДКРИТИМИ КЛЮЧАМИ

Найслабшою ланкою при реалізації симетричних криптосистем в системах захищеного електронного документообігу, електронних банківських платежів і, особливо, електронної торгівлі є питання розподілу ключів. Для вирішення цієї проблеми на основі результатів, одержаних класичною і сучасною алгеброю, були запропоновані системи з відкритим ключем (СВК), в яких для зашифрування не використовуються секретні ключі – вони потрібні тільки при розшифруванні (рис. 3.1) [1].

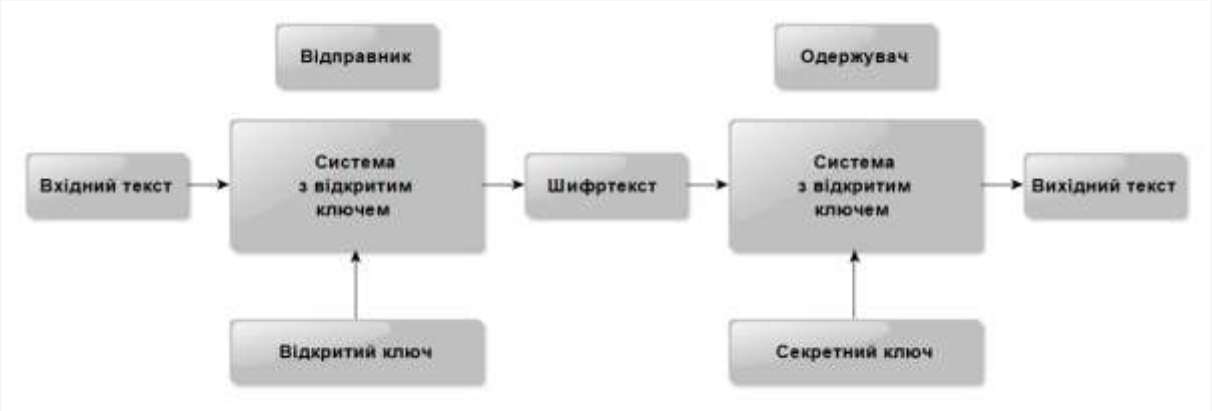


Рисунок 3.1 – Криптосистема з відкритим ключем

Важливим поняттям СВК є одностороння функція з секретом (one-way trap-door function) $f_t(x)$: $D \to R$, яку легко обчислити для всіх $x \in D$, проте для всіх значень із R надзвичайно складно виконати обернену задачу.

Криптосистеми з відкритим ключем, що використовують односторонню функцію з секретом, називають асиметричними (asymmetric cryptosystems) [6]. У таких криптосистемах інформація про відкритий ключ ніяк не допоможе відновити секретний, але наявність секретного ключа забезпечує можливість розшифрування повідомлення зашифрованого відкритим ключем.

У 1975 році У. Діффі та М. Хеллман запропонували принцип побудови асиметричних криптосистем. Крім цього, вони навели протокол формування секретного ключа – процедуру взаємодії по відкритому каналу зв'язку віддалених абонентів. Цей протокол базується на задачі дискретного логарифмування.

Найбільш відомими є асиметричні криптосистеми Меркла-Хеллмана, Рабіна, Шаміра, RSA, Ель-Гамаля, Діффі-Хеллмана та ін.

Варто зауважити, що на практиці часто застосовують комбіновані методи криптографії, де симетричні алгоритми використовуються для шифрування великих обсягів даних, а СВК – для створення механізмів розпо-

ділу ключів меншого розміру. Це поєднання забезпечує оптимальний баланс між стійкістю та ефективністю роботи криптоалгоритмів.

3.1 Криптографічна система RSA

Криптосистема RSA, що отримала свою назву від перших літер прізвищ її творців – Рона Рівеста (Ron Rivest), Аді Шаміра (Adi Shamir) та Леонарда Адлемана (Len Adleman), є найбільш поширеною і вивченою криптосистемою з відкритим ключем [9]. Алгоритм ґрунтується на тому, що легко перемножити два великих простих числа, але значно складніше розкласти їхній добуток на множники. У результаті добуток можна використати як елемент відкритого ключа шифрування.

Теорема Рабіна доводить, що розкриття шифру RSA еквівалентне знаходженню розкладу на прості множники великих чисел.

Розглянемо схему функціонування криптосистеми RSA.

1. Для генерації відкритого і секретного ключів користувачі вибирають два випадкових простих числа великої розрядності, p та q, й обчислюють їхній добуток [30]

$$n = p \cdot q. \tag{3.1}$$

Після цього обраховують значення

$$\varphi(n) = (p-1)(q-1). \tag{3.2}$$

Однією з вимог до вибору p та q є те, що принаймні одне з чисел $(p-1)$ або $(q-1)$ має мати один великий простий множник.

Крім того, розмір модуля n має бути не меншим, ніж 512 біт (в ідеалі довжина модуля має становити від 1024 біт).

2. Потім обирається ціле число e таке, що $e < \varphi(n)$ та $\text{НСД}\big(e, \varphi(n)\big) = 1$, тобто, числа e та $(p-1)(q-1)$ мають бути взаємно простими.

Відкритий ключ шифрування складається з пари чисел e, n.

3. Для знаходження секретного ключа d використовують розширений алгоритм Евкліда. Водночас, $ed = 1(mod(\varphi(n))$, звідки $d = e^{-1}mod(\varphi(n))$. Відмітимо, що числа d та n мають бути взаємно простими.

4. Перед початком шифруванням повідомлення m розбивають на блоки, довжини яких не перевищують n. Якщо потрібно зашифрувати фіксоване число блоків, то їх можна доповнити декількома нулями зліва для того, щоб гарантувати, що блоки завжди будуть меншими, ніж n. Зашифроване повідомлення c буде складатися з блоків c_i такої ж довжини [9].

Формула шифрування матиме вигляд

$$c_i = m_i^e mod\ n. \tag{3.3}$$

5. Для розшифрування повідомлення обирають кожен зашифрований блок c_i та обчислюють $m_i = c_i^d mod n$.

Процес шифрування та розшифрування за допомогою криптоалгоритму RSA відображено на рисунку 3.2.

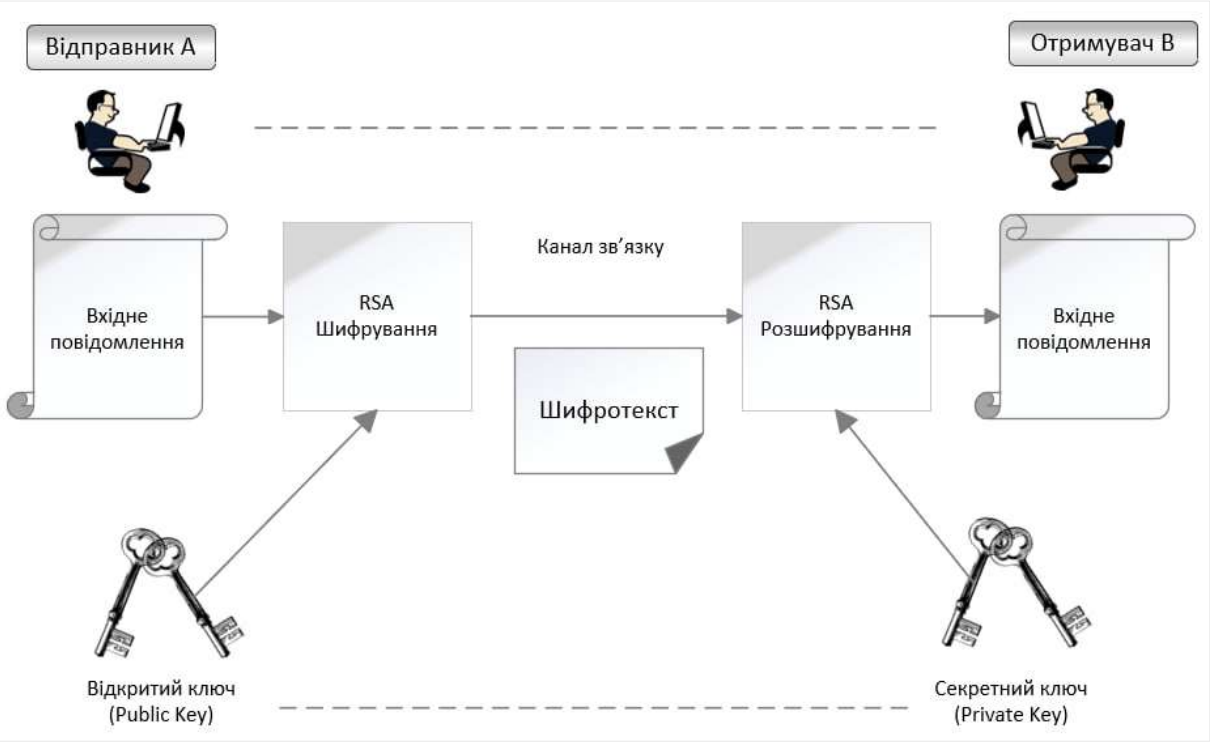


Рисунок 3.2 – Процес шифрування та розшифрування RSA

Розглянемо навчальний приклад, що демонструє роботу алгоритму RSA.

Нехай потрібно побудувати криптосистему RSA для $p = 5$, $q = 7$. Використовуючи перекодування алфавіту $А = 1$, $Б = 2$, $В = 3, \ldots$, $Я = 33$, зашифрувати повідомлення *ВЕСНА*, подавши результат у вигляді послідовності цифр [9].

1. Обчислимо $n = p \cdot q = 35$ та $\varphi(n) = (p-1)(q-1) = 24$.
2. Виберемо випадкове значення $e = 5$ таке, що НСД$\big(e, \varphi(n)\big) = 1$.

Отже, маємо відкритий ключ – (5, 35).

3. Обчислимо секретний ключ $d = 5$, $d = e^{-1} mod(\varphi(n))$.
4. Зашифруємо повідомлення: $m = 3\ 6\ 21\ 17\ 1$.

$RSA(\text{В}) = RSA\ (3) = 3^5 = 33\ mod 35$,
$RSA(\text{Е}) = RSA\ (6) = 6^5 = 6\ mod 35$,
$RSA(\text{С}) = RSA\ (21) = 21^5 = 21\ mod 35$,
$RSA(\text{Н}) = RSA\ (17) = 17^5 = 12\ mod 35$,
$RSA(\text{А}) = RSA\ (1) = 1^5 = 1\ mod 35$.

Отримуємо зашифроване повідомлення $\text{С} = 33\ 6\ 21\ 12\ 1$

5. Для розшифрування піднесемо кожний блок до степеня $d = 5$ за модулем 35:

$33^5 = 39135393 \equiv 3\ mod35,$

$6^5 = 7776 \equiv 6\ mod35,$

$21^5 = 4084101 = 21 mod35,$

$12^5 = 248832 = 17 mod35,$

$1^5 = 1 = 1 mod35$

В результаті розшифрування було одержано вихідне повідомлення $m = 3, 6, 21, 17, 1.$

Відобразимо даний приклад за допомогою таблиці 3.1.

Таблиця 3.1 – Побудова криптосистеми RSA навчального прикладу

Користувач А	Канал зв'язку	Користувач В
		$m = 3\ \ 6\ 21\ 17\ 1$
$n = 35$	$n = 35\ \rightarrow$	$n = 35$
$e = 5$	$e = 5 \rightarrow$	$e = 5$
$d = 5$		
$\mathrm{C} = 33\ 6\ 21\ 12\ 1$	$\leftarrow \mathrm{C} = 33\ 6\ 21\ 12\ 1$	$\mathrm{C}_1 = 3^5 mod35 = 33$ $\mathrm{C}_2 = 6^5 mod35 = \ 6$ $\mathrm{C}_3 = 21^5 mod35 = 21$ $\mathrm{C}_4 = 17^5 mod35 = 12$ $\mathrm{C}_5 = 1^5 mod35 = 1$
$m_1 = 33^5 mod35 = 3$ $m_2 = 6^5 mod35 = 6$ $m_3 = 21^5 mod35 = 21$ $m_4 = 12^5 mod35 = 17$ $m_5 = 1^5 mod35 = 1$		

Секретний ключ для навчальної системи легко знайти перебором. На практиці це неможливо, оскільки реальний розмір модуля (довжина бітового подання) $size(n)$ знаходиться в діапазоні від 512 до 4096 біт. Основні зусилля в ході практичної реалізації RSA припадають на генерацію випадкових великих простих чисел. Є очевидний шлях рішення задачі: випадково вибрати велике непарне число n і перевірити його подільність на множники в діапазоні $3, \ldots, \sqrt{n}$. У разі невдачі вибираємо число $n + 2$ і так далі. Проте при великих n такий підхід нездійсненний [1].

Стійкість криптосистеми RSA базується на складності розкладання модуля на два великих простих множники. Якби дану задачу можна було б розв'язати, тоді з легкістю обчислюється функція Ейлера від модуля, а потім, використовуючи алгоритм Евкліда, можна за відкритим ключем визначити секретний.

Одна з можливих атак на криптосистему RSA полягає в знаходженні $\varphi(n)$, оскільки в цьому випадку можна легко обчислити d із співвідношен-

ня $ed = 1(mod(\varphi(n))$. Проте наступні міркування доводять, що даний підхід не є простим. Якщо відомо значення $\varphi(n)$, то p та q можна знайти так. З рівності [31]

$$\varphi(n) = (p-1)(q-1) = pq - (p+q) + 1 = n - (p+q) + 1 \qquad (3.4)$$

видно, що за допомогою $\varphi(n)$ легко знаходиться $p+q$. Більш того, знаючи n та $p+q$, можна легко обчислити $p-q$, використавши співвідношення

$$(p-q)^2 = (p+q)^2 - 4pq = (p+q)^2 - 4n. \qquad (3.5)$$

Отже, якщо відомо $p-q$ та $p+q$, то p та q можна знайти за допомогою формул

$$p = (1/2)[(p+q) + (p-q)], \qquad (3.6)$$

$$q = (1/2)[(p+q) - (p-q)]. \qquad (3.7)$$

Цей факт доводить, що будь-яка спроба знайти функцію Ейлера від модуля n еквівалентна вирішенню складної проблеми розкладанню n на множники. Тобто, стійкість системи RSA до атаки на основі знаходження $\varphi(n)$ не є нижчою складності розкладання модуля на прості множники.

Можливість реально оцінити захищеність алгоритму RSA стала однією з причин популярності цієї СВК на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп'ютерних мережах, особливо для роботи з віддаленими клієнтами (обслуговування кредитних карток). В наш час алгоритм RSA використовується в багатьох стандартах, серед яких SSL, S-HTTP, S-MIME, S/WAN, STT і PCT. Крім того, алгоритм RSA реалізується як у вигляді самостійних криптографічних продуктів (PGP), так і як вбудовані засоби в деяких додатках (Інтернет-браузери від Microsoft і Netscape) [1].

3.2 Криптографічний протокол розподілу ключів Діффі-Хеллмана

У процесі створення секретного зв'язку системи надзвичайно важливим завданням є розробка системи керування ключами, яка охоплює обробку і передачу інформації.

Процес керування ключами містить три основні складові:

– генерацію ключів;

– зберігання ключів;

– розподіл ключів.

У практичних криптографічних системах для генерації випадкових ключів застосовуються спеціальні апаратні та програмні методи, які потребують наявності певного «випадкового фактора», наприклад, послідовності випадкових чисел.

У складній системі один користувач може оперувати значним обсягом ключової інформації, яка потребує організації спеціальних баз даних, що забезпечують прийом, зберігання, облік та видалення ключів.

Секретні ключі ніколи не мають записуватися в явному вигляді на носіях, які можуть бути прочитані або скопійовані. В той же час, інформація про ключі, які використовуються в системі, має зберігатися у зашифрованому вигляді. Зазначимо, що ключі, які шифрують ключову інформацію, називаються майстер-ключами.

Важливою умовою забезпечення безпеки інформації є періодичне оновлення ключової інформації в системі, яке пов'язано з розподілом ключів – найвідповідальнішим процесом у керуванні ключами.

До процесу розподілу ключів висуваються такі базові вимоги як оперативність й точність розподілу та недоступність розподілюваних ключів для сторонніх осіб.

Для обміну ключами досить ефективним виявився алгоритм на основі відкритих ключів, який був наведений у роботі У. Діффі та М. Хеллмана [2]. Суть цього алгоритму полягає у наданні двом користувачам можливості захищеного обміну ключами, які вони можуть згодом використовувати для шифрування повідомлень. Сам алгоритм обмежується процедурою обміну ключами.

У. Діффі та М. Хеллман запропонували використовувати односторонню функцію дискретного піднесення до степеня як основу для створення асиметричних криптосистем [24]. Отже, ефективність алгоритму Діффі-Хеллмана базується на складності обчислення дискретних логарифмів.

Протокол Діффі-Хеллмана здійснює експоненціальний ключовий обмін, з використанням функції $f(x)$, при заздалегідь вибраних несекретних параметрах g, p, які вибираються так, щоб послідовність степенів

$$g^1 mod p, g^2 mod p, \dots, g^{p-1} mod p$$

збігалася, з точністю до перестановки, з послідовністю $1,2, \dots p\ -1$.

Число g називається первісним коренем.

Для будь-якого цілого числа b та довільного первісного кореня g простого числа p однозначно визначається показник степеня i, при якому [31]:

$$b = g^i mod\ p, \text{де}\ 0 \le i \le p-1. \qquad (3.8)$$

Саме показник степеня є дискретним логарифмом.

Опишемо процес обміну ключами за схемою Діффі-Хеллмана, де наявні два відкритих числа: p (просте) та g (ціле), що є первісним коренем p.

Нехай користувачі А і В планують здійснити обмін ключами. Користувач А вибирає випадкове ціле число $X_A < p$ та обчислює $Y_A = g^{X_A} mod\ p$.

Аналогічно користувач В незалежно вибирає випадкове ціле число $X_B < p$ та обчислює $Y_B = g^{X_B} mod\ p$.

Обидві сторони зберігають значення X у таємниці, а значення Y роблять загальнодоступним.

Для знаходження ключа користувач А застосовує формулу $K = (Y_B)^{X_A} mod\ p$, а користувач В – формулу [31] $K = (Y_A)^{X_B} mod\ p$.

Причому, $K = (Y_B)^{X_A} modp = (g^{X_B} modp)^{X_A} modp = (g^{X_B})^{X_A} modp = = g^{X_B X_A} modp = (g^{X_A})^{X_B} modp = (g^{X_A} modp)^{X_B} modp = (Y_A)^{X_B} modp$

Так обидві сторони обмінялися секретним ключем. Оскільки при цьому X_A і X_B були тільки в особистому користуванні, то вони збереглися в таємниці, і зловмиснику доведеться працювати тільки з p, g, Y_A, Y_B. Отже, для визначення ключа йому доведеться обчислювати дискретний логарифм [24].

Графічна ілюстрація механізму обміну ключами за протоколом Діффі-Хеллмана відображена на рисунку 3.3.

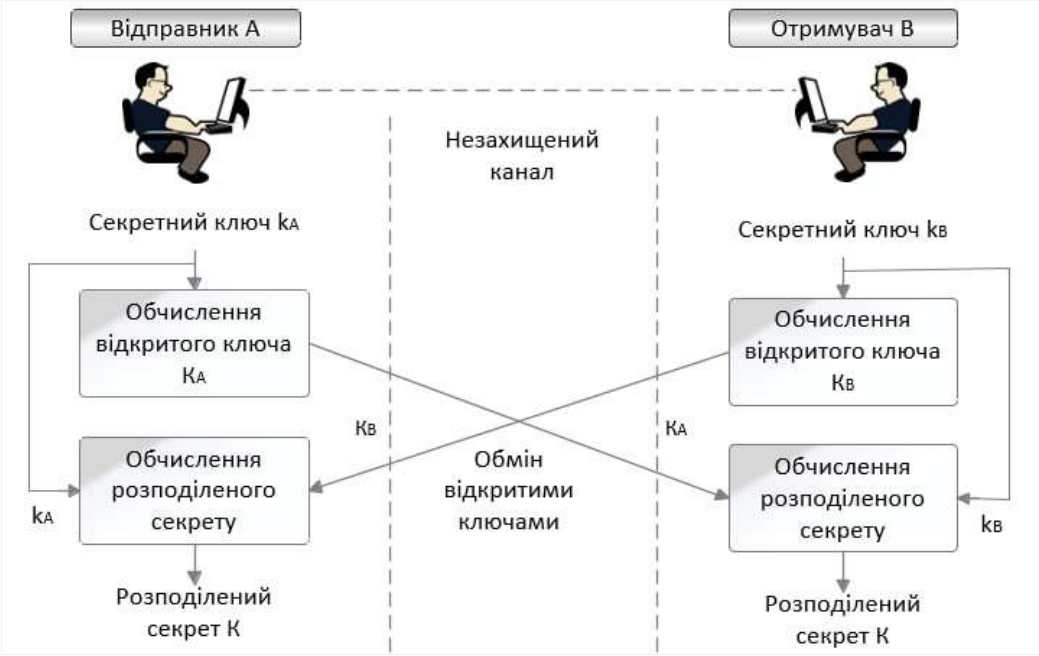


Рисунок 3.3 – Схема відкритого розподілу ключів Діффі-Хеллмана

Розглянемо приклад розподілу ключів згідно з алгоритмом Діффі-Хеллмана.

Побудуємо параметри для протоколу експоненціального ключового обміну при $q = 11$ та опишемо роботу протоколу для вибраних значень.

Обчислимо первісний корінь

$g = 7$, оскільки $\{7^1, 7^2, \dots, 7^{10}\} = \{7, 5, 2, 3, 10, 4, 6, 9, 8, 1\}$.

1. Користувач А генерує псевдовипадкове число $X_A = 5$ і передає користувачу В значення $Y_A = g^{X_A} mod\ p = 7^5 mod\ 11 = 10$.
2. Користувач В генерує псевдовипадкове число $X_B = 9$ і передає користувачу А значення: $Y_B = g^{X_B} mod\ p = 7^9 mod\ 11 = 8$.
3. Користувач А обчислює ключ

$$K = (Y_B)^{X_A} mod\ p = 8^{15} mod\ 11 = 10.$$

4. Так само користувач В також обчислює ключ

$$K = (Y_A)^{X_B} mod p = 10^9 mod\ 11 = 10.$$

Безпека протоколу Діффі-Хеллмана ґрунтується на асиметрії обчислювальних задач, а саме: піднесення до степеня за модулем великого простого числа виконується відносно легко, тоді як обернена операція – знаходження дискретного логарифма – є надзвичайно складною.

Розглянемо приклад іншої можливості використання алгоритму Діффі-Хеллмана.

Нехай кожен користувач локальної мережі має згенерувати секретне значення X_A для довгострокового застосування і обчислити відкрите значення Y_A. Усі отримані відкриті значення разом із глобальними відкритими значеннями p і g зберігаються централізовано в деякому каталозі.

У будь-який момент часу користувач В може отримати доступ до відкритого значення користувача А, обчислити секретний ключ і використати його для надсилання шифрованого повідомлення користувачеві А.

Якщо каталог, що централізовано зберігається, надійний, то ця форма зв’язку забезпечує як конфіденційність, так і певний ступінь автентифікації. Оскільки тільки користувачі А і В можуть визначити ключ, інший користувач не зможе прочитати повідомлення (конфіденційність). Одержувач А знає, що тільки користувач В міг створити повідомлення, використовуючи цей ключ (аутентифікація). Однак така схема не захищена від атак відтворення повідомлень.

Проблема аутентифікації відкритих ключів є дуже важливою. Коректність протоколів, які використовують асиметричне шифрування, може бути забезпечена тільки за умови, що всі відкриті ключі є справжніми. Якщо зловмиснику вдасться підмінити відкритий ключ будь-якого користувача, таємні повідомлення, надіслані цьому користувачу, будуть доступні зловмиснику. Таким чином, двоключові шифри вирішують проблему розподілу секретних ключів, але питання аутентифікації залишається, хоча вимоги підтвердження справжності (аутентифікації) стосуються вже відкритого, а не секретного ключа. В неявному вигляді аутентифікація відкритого ключа містить в собі аутентифікацію секретного ключа.

3.3 Криптографічна система Ель-Гамаля

У 1985 році американський криптограф Тахер Ель-Гамаль (Taher ElGamal) опублікував статтю «Криптосистема з відкритим ключем, схема

цифрового підпису на основі дискретних логарифмів» [32], у якій описав алгоритми асиметричних систем шифрування та створення електронного підпису.

Криптосистема Ель-Гамаля забезпечує як шифрування даних, так і створення цифрових підписів. Ця схема лежить в основі колишнього стандарту електронного цифрового підпису в США (DSA) [9].

Фактично такий шифр використовує протокол Діффі-Хеллмана для створення спільного секретного ключа, який потім застосовується для шифрування відкритого повідомлення. Секретний ключ не використовується повторно, а генерується заново для кожного нового повідомлення.

Основою безпеки алгоритму Ель-Гамаля є одностороння функція дискретного піднесення до степеня за простим модулем p. Секретні параметри системи визначають показник степеня.

Алгоритм Ель-Гамаля базується на складності обчислення дискретного логарифма в скінченному полі. Піднесення до степеня є легкою операцією, тоді як обернена операція – знаходження показника степеня – є обчислювально складною проблемою.

У криптосистемі Ель-Гамаля для побудови пари асиметричних ключів вибирається велике просте число p і два псевдовипадкові числа, менші $p-1$. Одне з них, g, має бути елементом великого порядку за модулем p, скажімо, первісним коренем. Друге число, x, вибирається як секретний ключ. Вважається, що повідомлення – лишки за модулем p. Відкритим ключем є трійка чисел $p, g, y = g^x \, mod \, p$, секретним ключем – число x. Для кожного повідомлення формуються додаткові дані, що відіграють роль лазівки для конкретного сеансу шифрування. Для зашифрування повідомлення M вибирається псевдовипадкове число k (рандомізатор, разовий ключ) з умовою НСД$(k, p-1) = 1$. Рандомізатори не мають повторюватися і мають триматися в секреті. Потім обчислюються числа $a = g^k \, mod \, p$ – лазівка і $b = y^k M mod \, p$ – шифротекст. Криптограмою є пара блоків даних a, b. Для розшифрування достатньо одержати співмножник y^k, що можна зробити за допомогою секретного ключа, обчисливши значення $a^x = g^{kx} \, mod \, p$. Дійсно, $y^k = g^{xk} \, mod \, p$, тому $M = b(a^x)^{-1} mod \, p$. Отже, забезпечується захищений інформаційний обмін без попереднього розсилання секретних ключів [1].

Схематично алгоритм Ель-Гамаля відображено на рисунку 3.4.

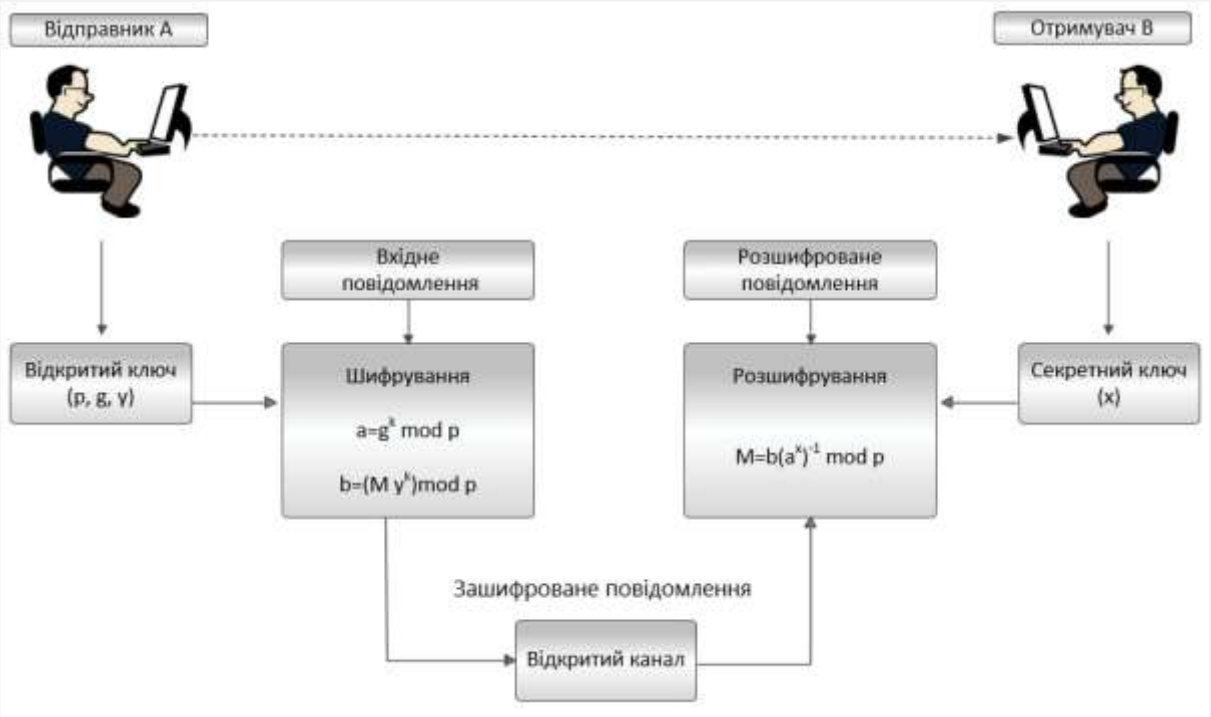


Рисунок 3.4 – Схема Ель-Гамаля

Розглянемо приклад шифрування та розшифрування повідомлення $M = 7$, використовуючи криптоалгоритм Ель-Гамаля [18].

1. Нехай $p = 23,\ g = 5$. Виберемо випадкове ціле число $x = 6$, $(1 < x < p)$.
2. Далі обчислимо $y = g^x mod p = 5^6 mod 23 = 8$.
3. Отже, відкритим ключем є $(p, g, y) = (23, 5, 8)$, а секретним – $x = 6$.
4. Нехай $k = 10$. Обчислимо число a

$$a = g^k mod p = 5^{10} mod 23 = 9.$$

5. Обчислимо число b

$$b = y^k M mod p = 8^{10} 7 mod 23 = 21.$$

6. Отримаємо такий шифротекст $(a, b) = (9, 21)$.

Процес розшифрування матиме нижчеописаний вигляд.

1. За відомим шифротекстом $(a, b) = (9, 21)$ та ключем $x = 6$ потрібно отримати повідомлення $M = 7$.
2. Знайдемо M за формулою

$$M = b(a^x)^{-1} mod p = 21 \cdot (9^6)^{-1} mod 23 = 7.$$

Оскільки шифрування за схемою Ель-Гамаля передбачає використання випадкового числа k, то цей алгоритм відносять до класу ймовірнісних шифрів. Завдяки цьому кожне шифрування одного і того ж повідомлення дає різний шифротекст, що ускладнює криптоаналіз. Проте схема

Ель-Гамаля має і недолік, який полягає у значному збільшенні розміру шифротексту порівняно з вихідним повідомленням. Адже суть відкритого шифрування Ель-Гамаля полягає у множенні за модулем повідомлення на разовий спільний секрет. Для того, щоб i-ий абонент міг правильно розшифрувати повідомлення, в криптограму разом з шифротекстом C додають разовий відкритий ключ відправника, що призводить до збільшення розміру криптограми приблизно в два рази порівняно з вихідним текстом.

Для схеми ймовірнісного шифрування саме повідомлення M і ключ не визначають шифротекст однозначно. У схемі Ель-Гамаля потрібно використовувати різні значення випадкової величини k для шифрування різних повідомлень M та $\acute{M}$. Якщо використовувати однакові k, то для відповідних шифротекстів (a, b) та $(\acute{a},\ \acute{b})$ виконується співвідношення [32]

$$b(\acute{b})^{-1} = M(\acute{M})^{-1}. \qquad (3.9)$$

Із цього виразу можна легко обчислити $\acute{M}$ якщо відомо M.

Отже, алгоритм Ель-Гамаля є надійним методом асиметричного шифрування, заснованим на обчислювальній складності задачі дискретного логарифмування. Завдяки своїм властивостям, шифр Ель-Гамаля забезпечує високий рівень безпеки, особливо при використанні великих ключів і відповідних криптографічних параметрів, що ускладнюють реалізацію різноманітних криптографічних атак.

3.4 Електронний цифровий підпис

Електронним цифровим підписом (ЕЦП) називається результат спеціального криптографічного перетворення, здійсненого над електронним документом його власником, з метою доведення незаперечності тексту документа і факту перетворення даних конкретною особою.

Формуючи протокол ЕЦП потрібно:

– створити пару повідомлення/підпис таким чином, щоб її неможливо було підробити;

– уміти здійснити перевірку того, що ЕЦП дійсно належить вказаному власнику.

Крім того, ЕЦП потрібно створювати так, щоб підписант повідомлення з часом не міг заперечити факт підписання, стверджуючи, що підпис сфабрикований.

Загалом ЕЦП являє собою деяке число специфічної структури, яке допускає перевірку за допомогою відкритого ключа того факту, що воно було вироблено для деякого повідомлення з використанням секретного ключа. Таким чином у схемі цифрового підписування відправник генерує два ключі – загальнодоступний відкритий ключ K_1 та відповідний йому

секретний ключ K_2. При формуванні підпису для повідомлення M відправник обчислює цифровий підпис DS від M, використовуючи ключ K_2. При перевірці підпису одержувач перевіряє підпис DS від повідомлення M , використовуючи ключ K_1 [33].

Виходячи з цього, варто відзначити, що процес цифрового підписування базується на двох взаємодоповнювальних операціях: генерації підпису та його верифікації.

Існує широкий спектр схем цифрового підписування, серед яких можна виділити:

– схеми підписування з додаванням (with appendix). Цифровий підпис DS приєднується до повідомлення M і у такому вигляді відправляється одержувачу, а під час перевірки цього підпису необхідно мати підпис DS та саме повідомлення M;

– схеми підписування з відновленням повідомлення (with message recovery). Все повідомлення (або його частина) може бути відновлено безпосередньо з цифрового підпису, тому процедура перевірки підпису оперує виключно з наданим цифровим підписом DS, не вимагаючи додаткової інформації.

Кожна з цих схем цифрового підписування може бути:

– детермінованою (deterministic). При подаванні на вхід однакових повідомлень формуються однакові цифрові підписи, які, у свою чергу, поділяють на: схеми одноразового (one-time) застосування та схеми багаторазового (multiple-use) застосування;

– рандомізованою (randomized). Завдяки використанню певного випадкового числа (сеансового ключа) формуються різні цифрові підписи для однакових вхідних повідомлень (у разі використання тих самих ключів K_1 та K_2).

Серед існуючих схем цифрового підписування найбільшого поширення отримали рандомізовані схеми з додаванням повідомлення. До таких схем насамперед відносять методи RSA, Рабіна, Фейге-Фіата-Шаміра, Гіллоу-Куіскуотера, Шнорра, Ель-Гамаля, DSA, що базуються на математичному апараті піднесення до степеня великих цілих чисел у скінченому полі [33], а також методи на основі математичного апарата еліптичних кривих над скінченими полями, зокрема ECDSA, ДСТУ 4145-2002 та ін.

У процесі формування ЕЦП за класичною схемою відправник повідомлення спочатку застосовує до вихідного тексту хеш-функцію, а потім обчислює цифровий підпис за допомогою секретного ключа створення підпису (рис. 3.5).

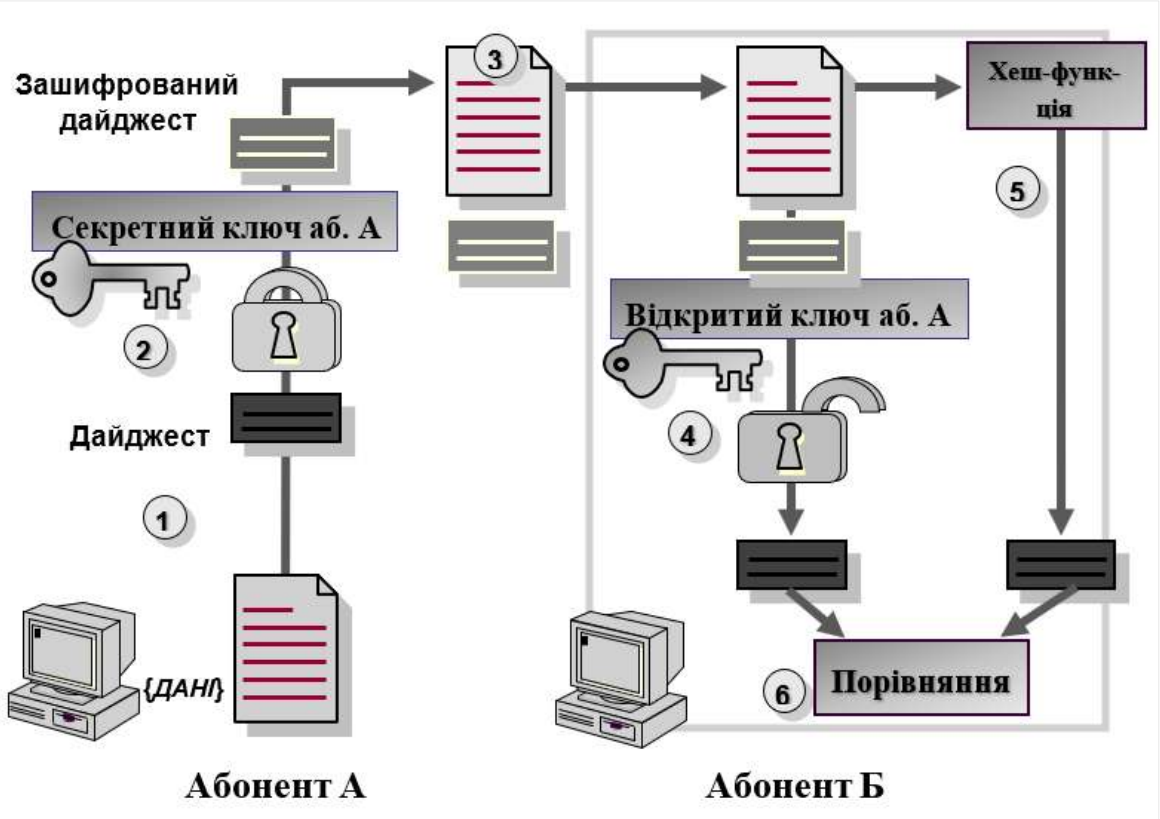


Рисунок 3.5 – Схема ЕЦП без шифрування повідомлення

Отримувач повідомлення виконує перевірку підпису: спочатку обчислює хеш-образ отриманого повідомлення за допомогою тієї ж самої хеш-функції, яку використовував відправник, а потім, за допомогою відкритого ключа, перевіряє чи відповідає цей хеш-образ отриманому ЕЦП.

3.4.1 Схема цифрового підпису RSA

Асиметричний криптоалгоритм RSA є найпопулярнішим, тому він якнайкраще ілюструє базові принципи побудови систем цифрового підпису.

Припустимо, що задана криптосистема RSA з параметрами $e,\ d,\ n$. Нехай абонент А має передати абоненту Б повідомлення m. Для цього абонент А, використовуючи свій секретний ключ d_A і відкритий ключ e_B абонента Б, «підписує» повідомлення за допомогою перетворення: $E_{e_B,n_B}\left(E_{d_A,n_A}(m)\right)$. Для перевірки підпису абонент Б спочатку розшифровує підписане повідомлення за допомогою свого секретного ключа d_B і одержує $E_{d_A,n_A}(m)$. Потім, використовуючи відкритий ключ абонента А, він отримає m, виходячи з $E_{d_A,n_A}(m)$ [1].

Автентичність повідомлення та підпису гарантується використанням приватного ключа користувача d_A. Однак існує можливість підробки випадкових даних, що є суттєвим недоліком для асиметричних систем, основне завдання яких – безпечна передача ключів.

Спробуємо нав'язати ключ m абоненту Б від імені абонента А, для чого, відповідно до протоколу, відішлемо абоненту Б повідомлення m виду $m = k^{e_A} \, mod \, n_A$, де k – довільний великий лишок за модулем n. Для цього потрібно зашифрувати повідомлення секретним ключем d_A і потім перешифрувати результат відкритим ключем e_B абонента Б. Проте ми знаємо результат зашифрування секретним ключем d_A $m^{d_A} = k^{e_A d_A} = k \, mod \, n_A$, і, отже, можемо перешифрувати результат ключем e_B. Таким чином, абонент Б отримає повідомлення $k = m^{d_A} \, mod \, n_A$, потім застосує відкритий ключ e_A абонента А і отримає $k^{e_A} = m \, mod \, n_A$ [1].

Аби убезпечитись від описаної атаки, потрібно застосувати комбінований метод шифрування. Спочатку повідомлення m шифрується відкритим ключем одержувача (якщо це потрібно). Потім секретний ключ відправника використовується для шифрування повідомлення $h = h(m)$ [1]. Функція h має бути складною та не оберненою.

Цифровий підпис RSA базується на парі $(m, c) = (m, h(m)^{d_A} \, mod \, n_A)$, де $h(m)$ – хеш-функція повідомлення m (відкрите, криптостійке перетворення вхідних даних будь-якого розміру у вихідний рядок фіксованої довжини).

Результат застосування хеш-функції $h(m)$ до повідомлення m називається хеш-кодом, причому довжина бітового подання хеш-коду є фіксованою і не залежить від довжини самого повідомлення.

Процес верифікації підпису (m, c) починається з обчислення хешу повідомлення m і подальшого порівняння результату з $c^e (mod \, n_A)$.

3.4.2 Протокол цифрового підписування Фіата-Шаміра

У 1986 році Амос Фіат та Адім Шамір запропонували схему ЕЦП, яка гарантує автентичність і цілісність повідомлень.

Протокол цифрового підписування Фіата-Шаміра базується на ідеї інтерактивних доведень нульового знання, що дозволяють відправнику довести автентичність свого підпису без розкриття секретного ключа. Цей протокол використовує криптографічні властивості великих простих чисел і їх добутку для генерації ключів, а також містить випадкові числа для забезпечення унікальності кожного підпису.

У схемі Фіата-Шаміра спільним відкритим ключем є число $n = p * q$; особистим секретним ключем – набір елементів $s_0, \dots, s_{k-1}$ кільця Z/nZ; особистим відкритим ключем – набір елементів $v_0, \dots, v_{k-1}$ кільця Z/nZ таких, що $v_i \equiv s_i^{-2} (mod n)$. Опишемо протокол Фіата-Шаміра [35].

На вході відправник має число n та $s_0, \dots, s_{k-1} \in Z/nZ$, а отримувач – n та $v_0, \dots, v_{k-1} \in Z/nZ$, де $v_i \equiv s_i^{-2} (mod n)$.

Для формування ЕЦП для повідомлення m довільної довжини відправник виконує такі кроки.

1. Вибирає довільне ціле число r, $1 < r < n - 1$.
2. Знаходить $u \leftarrow r^2 (mod n)$.

3. Обчислює хеш-функцію $e \leftarrow h(m \parallel u)$ і подає отримане значення у вигляді вектора $(e_0, \dots, e_{k-1})$.

4. Обчислює $s \leftarrow r \prod_{i=0}^{k-1} s_l^{e_l} \, (mod n)$.

Пара (e, s) є підписом для повідомлення m.

Для перевірки ЕЦП отримувач виконує такі кроки.

1. Подає число e у вигляді вектора $(e_0, \dots, e_{k-1})$.
2. Обчислює $w \leftarrow s^2 \prod_{i=0}^{k-1} v_i^{e_i} (mod n)$.
3. Знаходить $\acute{e} \leftarrow h(m \parallel w)$.
4. Перевіряє рівність $e = \acute{e}$.

Якщо рівність виконується, то підпис справжній, в іншому випадку – підпис недійсний.

Безпека цієї схеми підписування базується на складності добування кореня в кільці Z/nZ. Водночас число n має формуватися довіреною стороною. Крім того, в криптосистемі потрібно передбачити процедуру доведення того, що число n є добутком двох різних простих чисел.

Підписування методом Фіата-Шаміра не має обчислювальних морфізмів, властивих підпису RSA. Використання стійкої хеш-функції унеможливлює атаку, пов'язану з підробкою підпису.

Протокол вразливий до атак типу «людина посередині» (Man-in-the-Middle), що виникають, коли зловмисник може перехоплювати та підмінять повідомлення між підписувачем і верифікатором.

3.4.3 Алгоритм побудови електронного цифрового підпису Ель-Гамаля

У цьому розділі було проаналізовано схему шифрування Ель-Гамаля. Проте даний алгоритм може використовуватися і для формування ЕЦП. За такої умови генерація параметрів і вибір секретного і відкритого ключів відбувається як і у випадку шифрування. Різниця полягає в тому, що ЕЦП формується з використанням не відкритого ключа отримувача, а секретного ключа самого відправника.

Продемонструємо алгоритм побудови ЕЦП Ель-Гамаля [32].

1. За допомогою хеш-функції знаходиться дайджест $h(M)$ для повідомлення M.
2. Для генерації ключів вибираються: просте число p, число g – первісний корінь за модулем p та x– секретний ключ так, що $g < p$ і $x < p$.
3. Потім обчислюється відкритий ключ $y = g^x mod p$.
4. Далі генерується випадкове число k, $1 < k < p - 1$, яке має зберігатися в секреті і не має дублюватися для різних підписів.
5. Обчислюється $r = g^k mod p$.
6. Після чого розраховується $s = k^{-1}(h(M) - xr) mod (p - 1)$.

Пара (r, s) є ЕЦП для повідомлення M.

Для перевірки підпису потрібно використати відкриті параметри (p, g, y) та переконатися, що виконується рівність

$$y^r * r^s mod\ p = g^{h(M)} mod\ p. \quad (3.10)$$

Схема ЕЦП Ель-Гамаля відображена на рисунку 3.6.

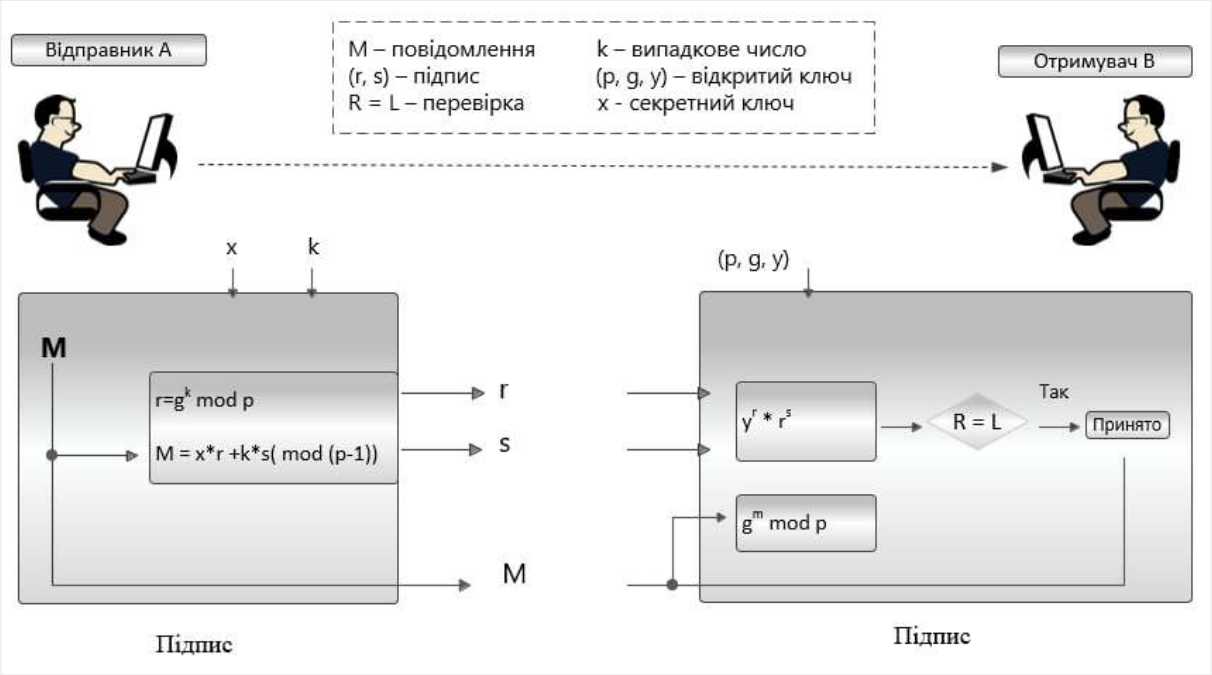


Рисунок 3.6 – Схема ЕЦП Ель-Гамаля

Вибір хеш-функції є важливою частиною цього алгоритму, оскільки від нього залежить криптографічна стійкість отриманої системи. Тому, вибираючи алгоритм хешування, мають враховувати такі параметри:

– складність обчислення;
– швидкість обчислення,
– широкий розкид хеш-значень, який зменшує кількість колізій.

Секретний ключ x, який використовується для обчислення елемента відкритого ключа y, що використовується надалі для підтвердження підпису, генерується випадковим чином у заданому діапазоні. Виходячи з цього, можна сказати, що використання некриптостійких генераторів псевдовипадкових чисел (ГПВЧ) призводить до підвищення ризику підробки цифрового підпису. Для перевірки стійкості ГПВЧ можна використовувати ентропійний аналіз, заснований на обчисленні міри невизначеності, яка показує непередбачуваність появи значень.

Існує багато різних модифікацій методу цифрового підписування Ель-Гамаля. Зокрема, метод Егню-Муллін-Ванстон, метод Йен-Лай, метод Нюберга-Рюпеля, Хорстера-Петерсена та інші.

3.5 Практичні аспекти розділу 3

3.5.1 Реалізація протоколу розподілу ключів Діффі-Хеллмана

Застосування криптографічного протоколу розподілу ключів Діффі-Хеллмана дає змогу проводити безпечний обмін криптографічними ключами через незахищений канал. За цих обставин учасники обміну можуть створити спільний секретний ключ для шифрування даних, який неможливо розкрити навіть у разі повного перехоплення їхньої комунікації зловмисником.

Для кращого розуміння роботи протоколу розподіл у ключів Діффі-Хеллмана відкриємо діалогове вікно CrypTool 2 «Diffie-Hellman Demonstration» (рис. 3.7) [10].

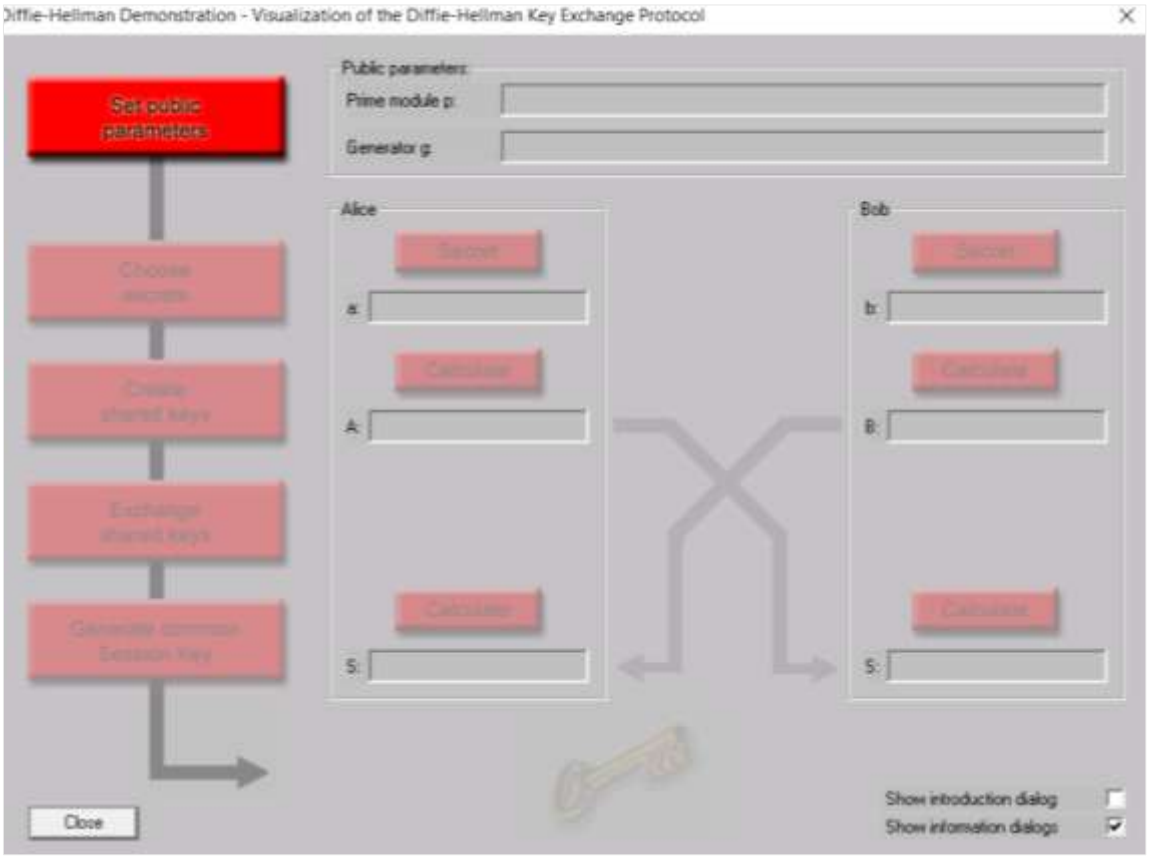


Рисунок 3.7 – Головне вікно «Diffie-Hellman Demonstration»

Припустимо, що дві сторони-учасники, Аліса і Боб, мають узгодити секретний ключ. Для безпечного обміну секретним ключем вони використовують протокол обміну ключами Діффі-Хеллмана.

Аліса і Боб вибирають публічні числа: просте число p та ціле число g, що є первісним коренем p.

Для демонстрації роботи цього протоколу виберемо невеликі числа. Нехай $p = 43$.

Для швидкого знаходження всіх можливих значень первісних коренів запустимо модуль «World of Primes».

Первісний корінь за модулем простого числа p є цілим числом g у $Zp = \{0, 1, \dots, p-1\}$ так, що кожен ненульовий елемент Zp є степенем g.

Отримаємо дванадцять первісних коренів для $p = 43$: $\{3, 5, 12, 18, 19, 20, 26, 28, 29, 30, 33, 34\}$ (рис. 3.8).

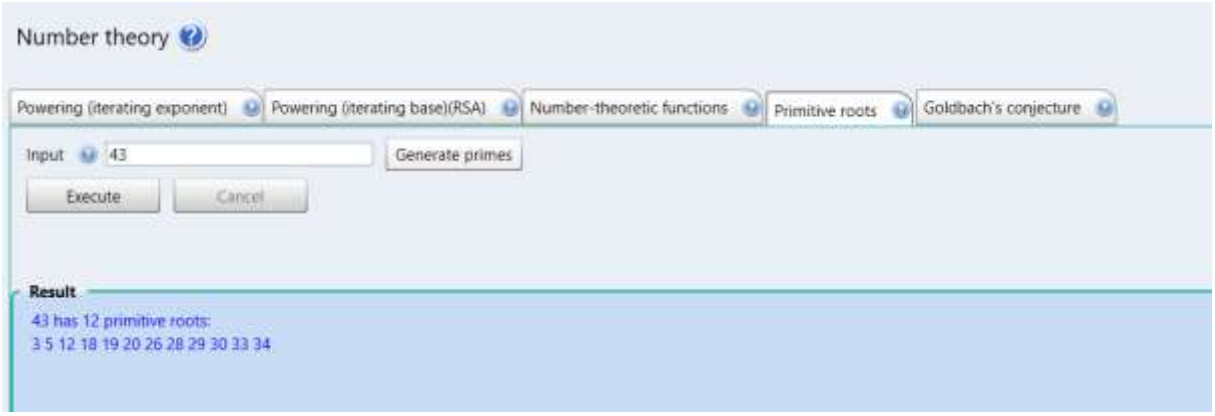


Рисунок 3.8 – Первісні корені для p

Первісний корінь за модулем m – це спеціальний елемент простих залишків з властивістю, що кожен з простих залишків може бути поданий у вигляді степеня первісного кореня.

Для прикладу виберемо g=3, оскільки g є примітивним коренем простих залишків числа 43, що належить проміжку від 1 до 42 (рис. 3.9).

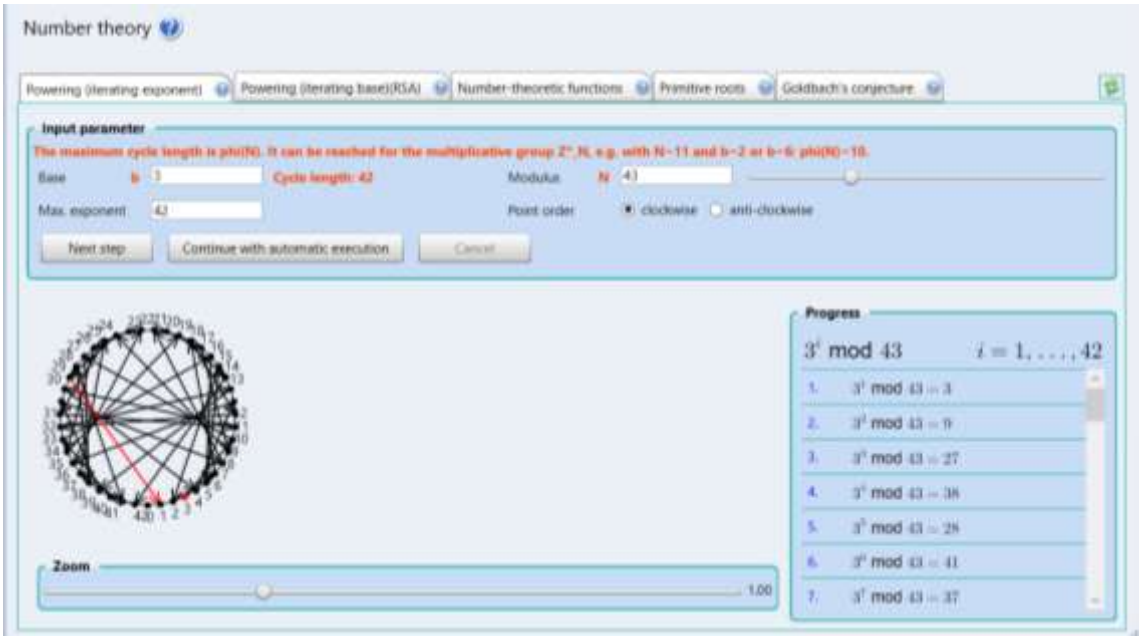


Рисунок 3.9 – Вибір первісного кореня g

Довжина циклу залежить від вибраних основи g та модуля p.

Для введення публічних параметрів p та g вибираємо «Set public parameters» і вводимо $p = 43$ та g= 3 (рис. 3.10).

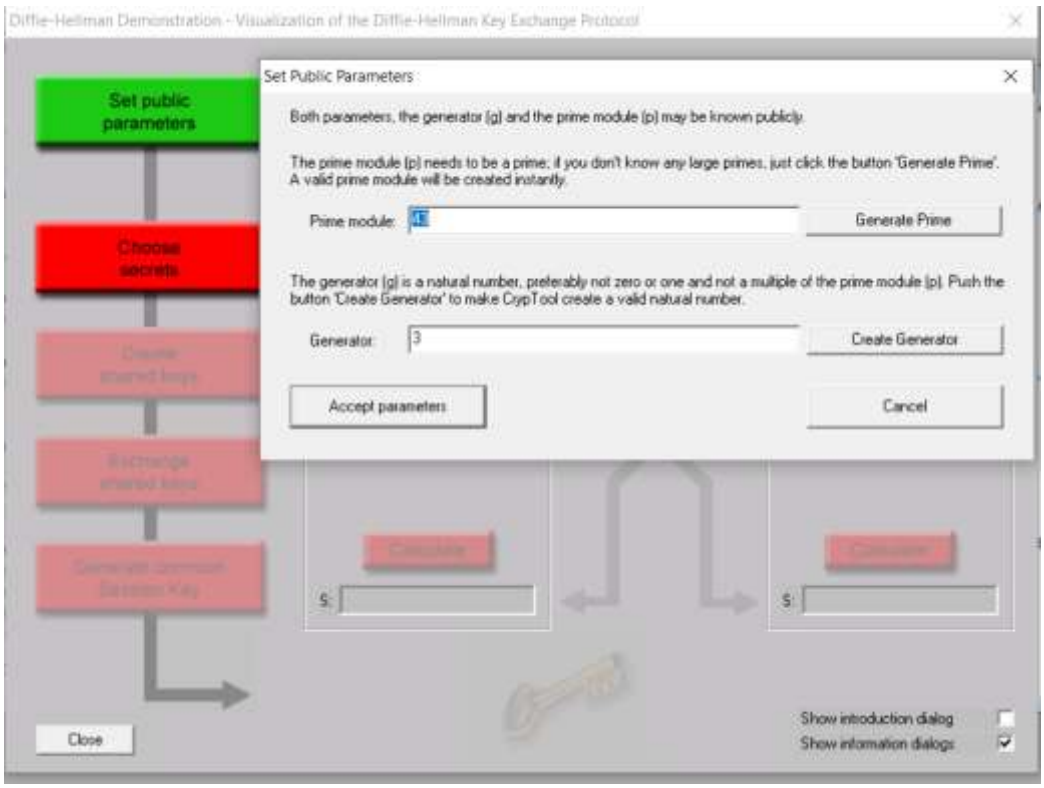


Рисунок 3.10 – Вибір публічних параметрів p та g

Наступним кроком є вибір Аліси випадкового цілого числа a, $1 < a < p - 1$, яке є секретним («Choose secrets»). Аліса вибирає ціле число $a = 7$ (рис. 3.11).

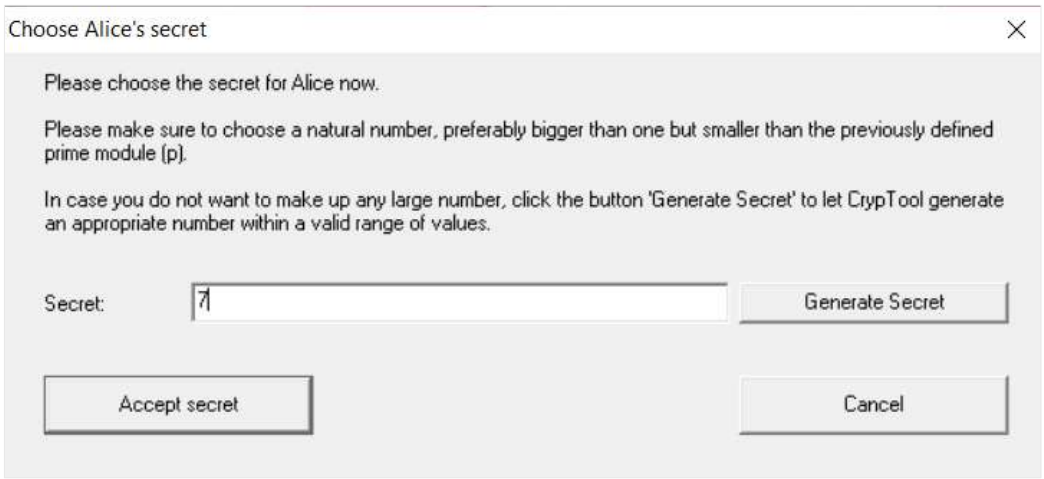


Рисунок 3.11 – Вибір секретного ключа a

Боб вибирає ціле число $b = 29$, яке він також тримає в секреті (рис. 3.12).

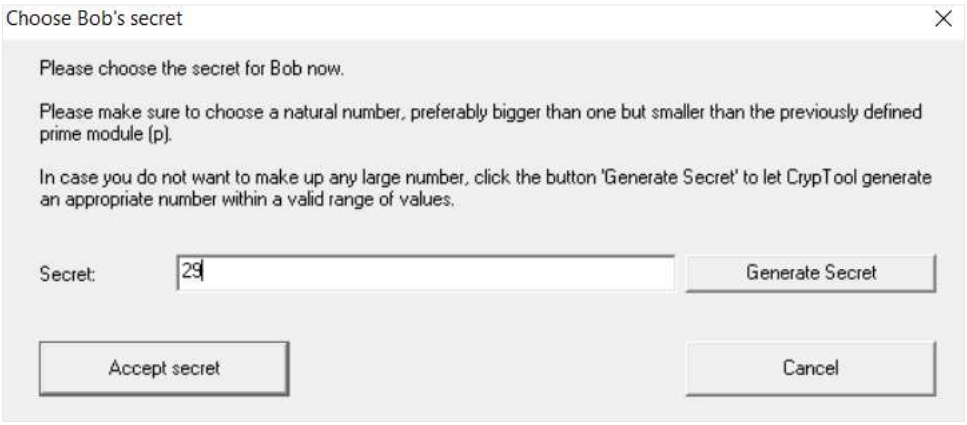


Рисунок 3.12 – Вибір секретного ключа b

Наступним кроком після вибору секретних ключів для Аліси (a) та для Боба (b) є обчислення власних відкритих ключів («Create shared keys»).

Аліса обчислює свій відкритий ключ за допомогою формули

$$A = g^a mod\ p = 3^7 mod\ 43 = 37.$$

Число a має зберігатися в секреті; однак A, відкритий ключ Аліси, надсилається Бобу.

Боб обчислює свій відкритий ключ B та надсилає його Алісі (рис. 3.13)

$$B = g^b mod\ p = 3^{29} mod\ 43 = 18.$$

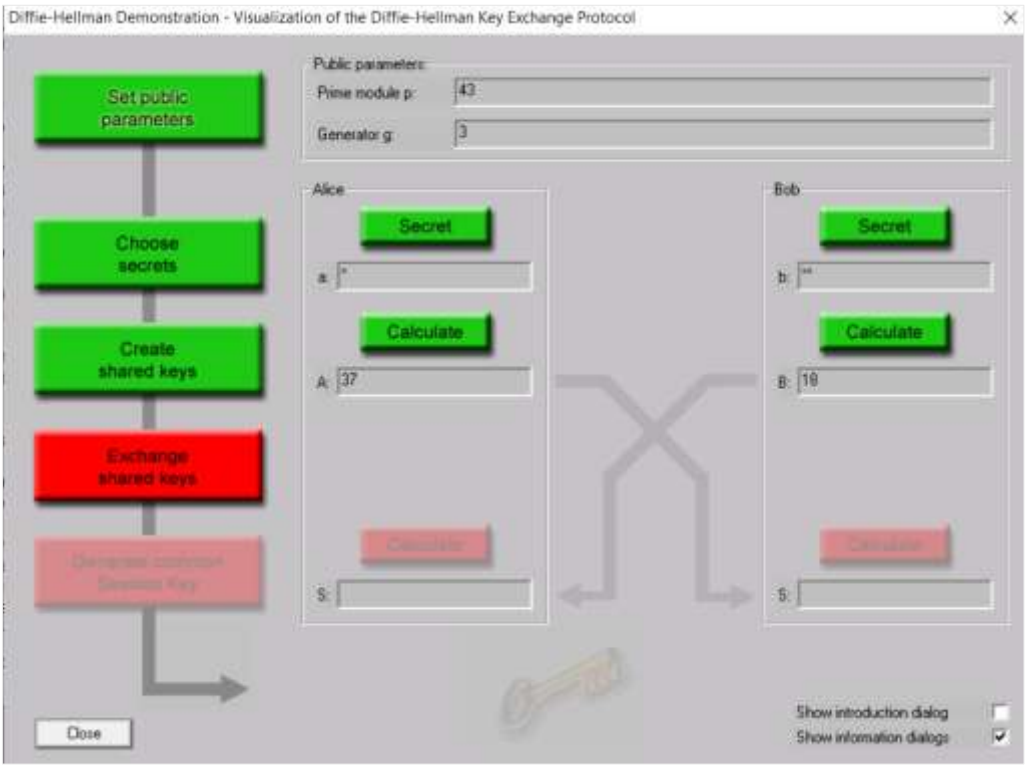


Рисунок 3.13 – Відкриті ключі Аліси та Боба

Після того як Аліса і Боб обчислили свої відкриті ключі Аліса надсилає свій відкритий ключ Бобу, а Боб – свій ключ Алісі. Для обміну ключами застосовуємо «Exchange shared keys» (рис. 3.14).

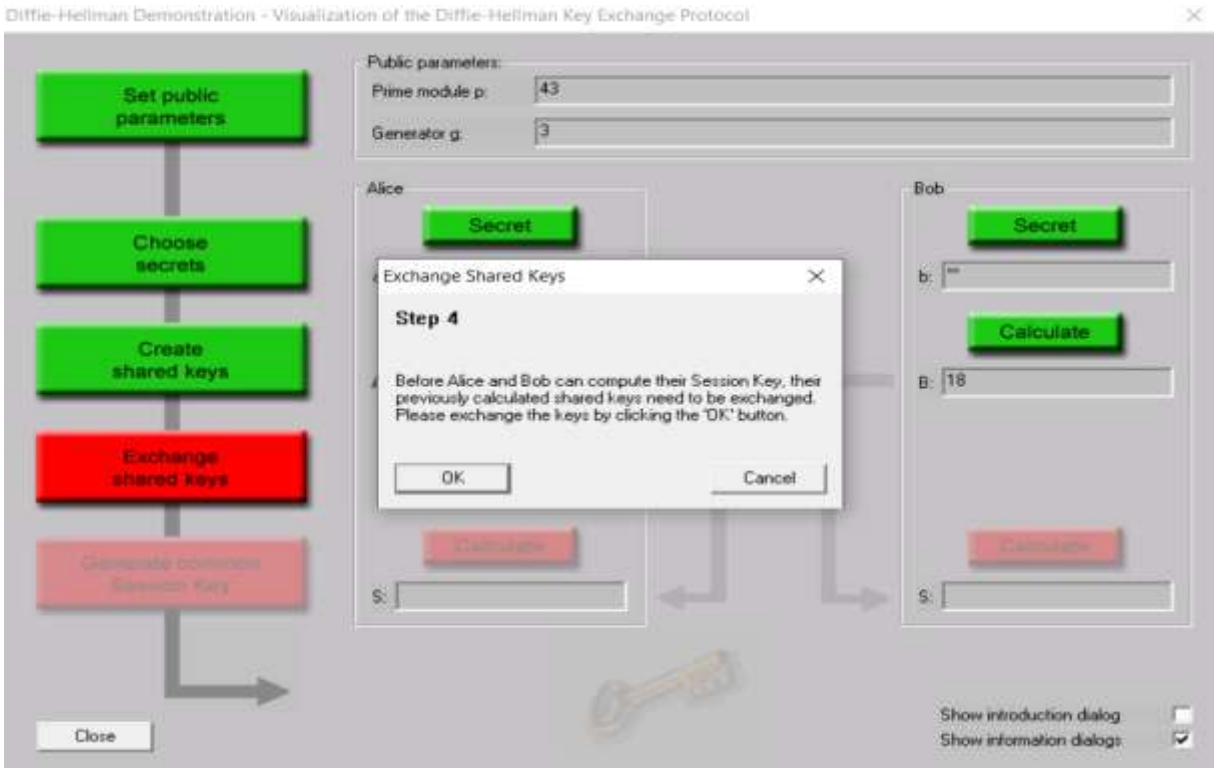


Рисунок 3.14 – Обмін спільними ключами

Останнім кроком є генерація секретного сеансового ключа («Generate common Session Key»).

Аліса обчислює загальний сеансовий ключ S

$$S = B^a mod\ p = 18^7 mod\ 43 = 7$$

Водночас Боб також обчислює загальний сеансовий ключ S

$$S = A^b mod\ p = 37^{29} mod\ 43 = 7$$

Отже, Аліса і Боб тепер згенерували спільний секретний сеансовий ключ $S = 7$, який можна використовувати для подальшого безпечного спілкування (рис. 3.15).

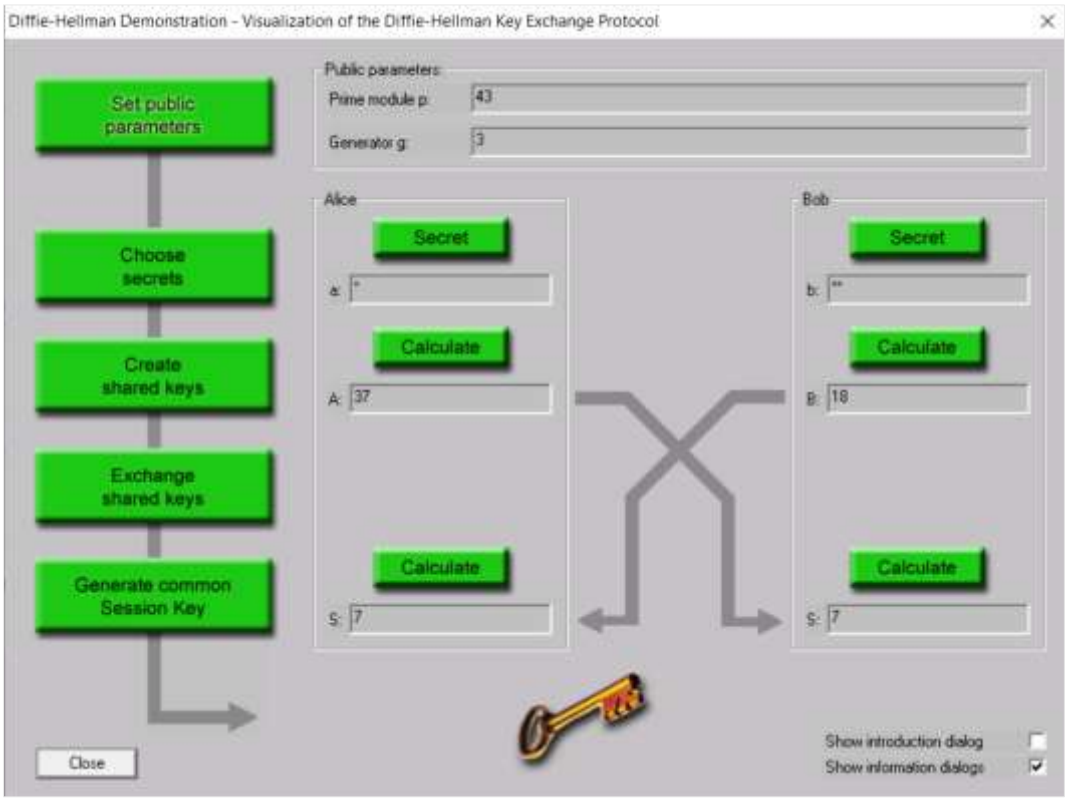


Рисунок 3.15 – Генерація спільного сеансового ключа

Отже, спільний сеансовий ключ є кінцевим результатом процесу обміну ключами, який використовується для конфіденційного обміну даними через загальнодоступну мережу під час зв’язку між двома сторонами. Цей спільний секретний ключ залишається невідомим іншим і забезпечує безпечний зв’язок.

Алгоритм Діффі-Хеллмана дозволяє двом користувачам встановити загальний секретний ключ через незахищений канал без передачі ключа.

Проблема вирішення логарифмічних обчислень у скінченному полі робить практично недосяжним визначення спільного секретного ключа без закритих ключів, що забезпечує процес обміну.

3.5.2 Створення цифрового підпису на основі асиметричного алгоритму RSA

Gpg4win (GNU Privacy Guard for Windows) – це безкоштовний програмний пакет, призначений для захисту електронної пошти та файлів за допомогою шифрування й цифрового підпису, що гарантує конфіденційність і цілісність даних [36]. Програма використовує криптографічні алгоритми на основі стандарту OpenPGP і забезпечує засоби для безпечного обміну даними, захищаючи їх від несанкціонованого доступу.

Gpg4win пропонує як графічний інтерфейс для зручної роботи з криптографічними операціями (через Kleopatra), так і можливість працювати через командний рядок (для досвідчених користувачів та автоматизації процесів).

Kleopatra – це графічний інтерфейс для керування сертифікатами та ключами, що є частиною програмного пакета Gpg4win [36]. Він дозволяє

користувачам створювати, імпортувати, експортувати та керувати криптографічними ключами для шифрування та підпису повідомлень і файлів за допомогою стандартів OpenPGP та S/MIME. Kleopatra також полегшує роботу з цифровими підписами та шифруванням даних.

Після запуску Kleopatra створимо пару ключів. Для цього виберемо опцію «Створити пару ключів» (рис. 3.16).

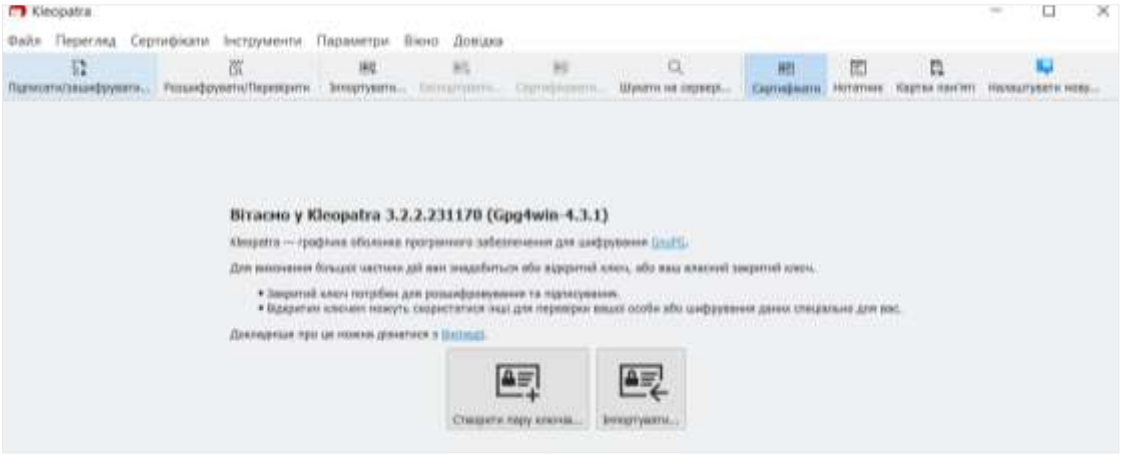


Рисунок 3.16 – Стартове вікно Kleopatra

Публічний ключ надається всім охочим, а приватний зберігається у надійному місці і служить для підписання інформації від імені його власника і розшифрування адресованої йому інформації [37].

Для створення пари ключів потрібно вказати ім’я та електронну адресу. Для захисту свого приватного ключа виберемо «Захистити створений ключ паролем» (рис. 3.17).

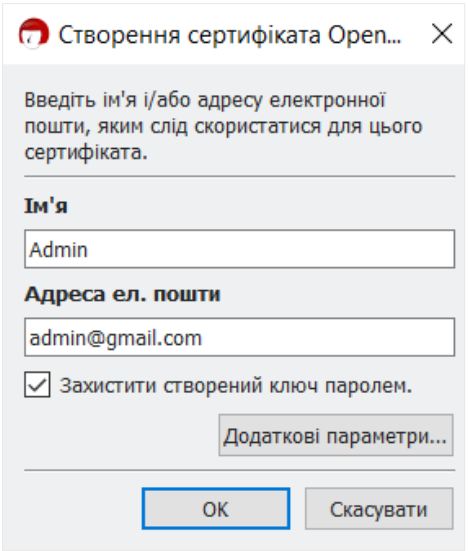


Рисунок 3.17 – Вікно для введення даних для створення ключів

Для створення ключів перейдемо в «Додаткові параметри» (рис. 3.18).

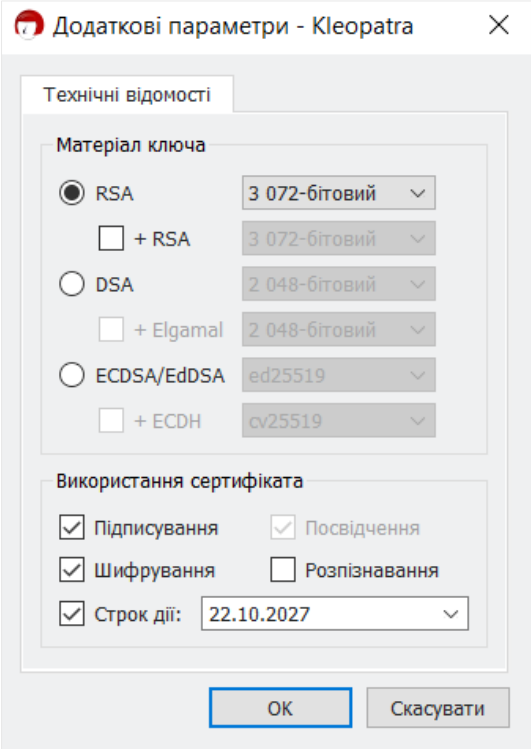


Рисунок 3.18 – Вибір алгоритму цифрового підпису

Додаткові параметри дозволяють визначити такі специфічні характеристики ключа, як тип криптографічного алгоритму, розмір ключа та термін дії. Вибираємо RSA 3072-бітовий з зазначеним строком дії.

Після введення пароля натискаємо «OK» та отримаємо повідомлення про успішне створення ключів та накладений відбиток (рис. 3.19).

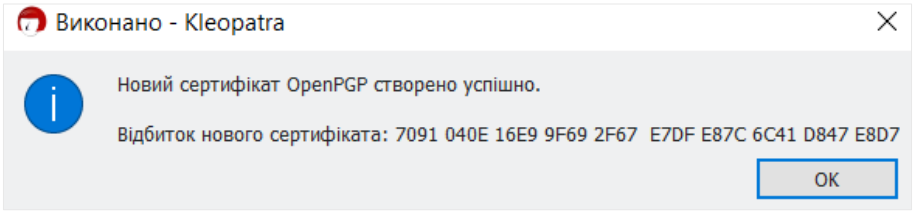


Рисунок 3.19 – Повідомлення про успішне створення ключів

Відбиток сертифіката – це унікальний ідентифікатор ключа. Навіть якщо хтось захоче представитися авторизованим користувачем, створивши ключ з аналогічним ім’ям та електронною поштою, його відбиток буде відрізнятися. Саме тому важливо зберігати відбитки отримуваних ключів.

В Kleopatra є можливість створення резервної копії приватного ключа, що потребує введення пароля (рис. 3.20).

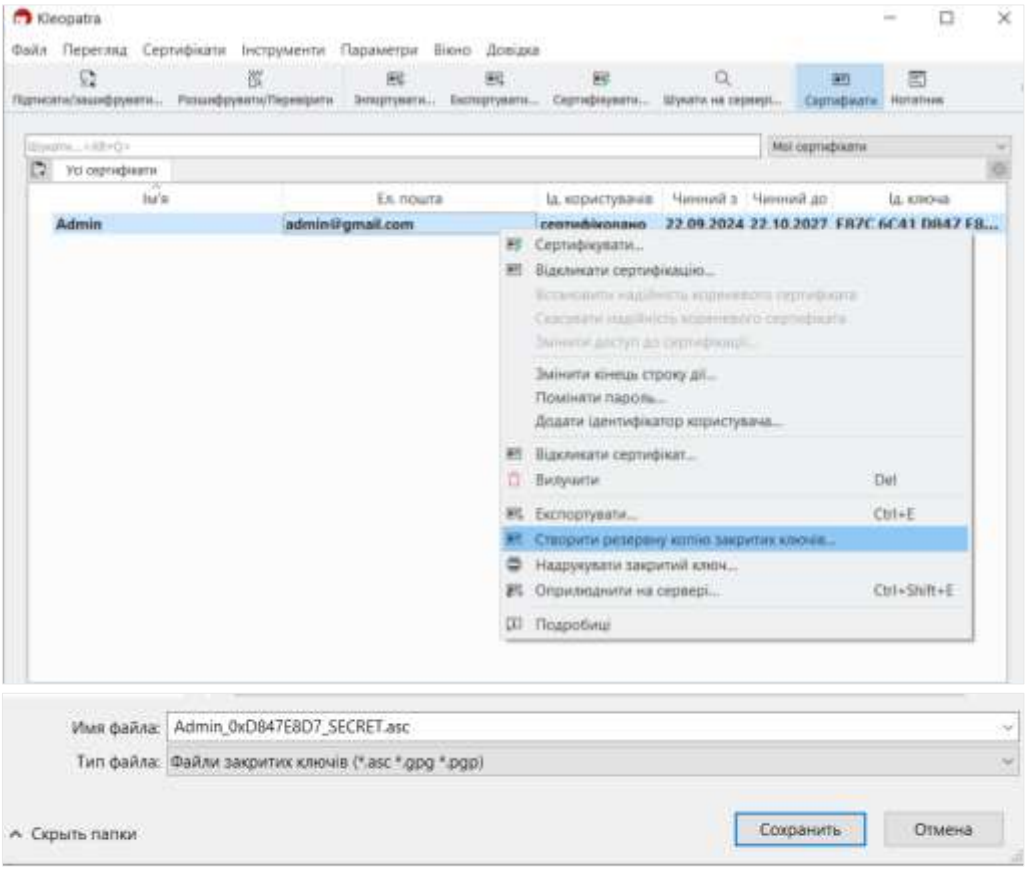


Рисунок 3.20 – Створення резервної копії приватного ключа

Після збереження резервної копії отримаємо повідомлення про успішне її створення (рис. 3.21).

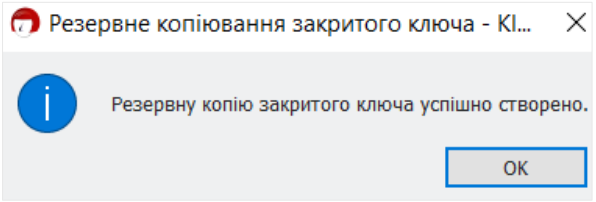


Рисунок 3.21 – Повідомлення про успішне створення резервної копії приватного ключа

Відкривши резервну копію ключа в будь-якому текстовому редакторі, побачимо «BEGIN PGP PRIVATE KEY BLOCK»[37] на початку тексту (рис. 3.22).

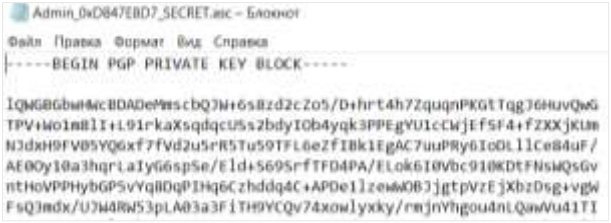


Рисунок 3.22 – Резервна копія приватного ключа

Для експорту публічного ключа вибираємо «Експортувати…» _Публічний ключ зберігається в форматі .asc (рис. 3.23).

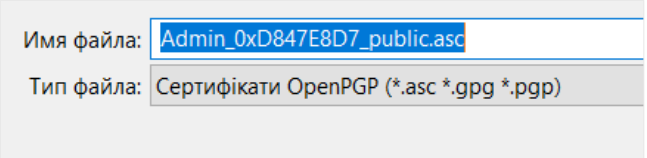


Рисунок 3.23 – Експорт публічного ключа

Відкривши файл побачимо «BEGIN PGP PUBLIC KEY BLOCK» на початку тексту (рис. 3.24).

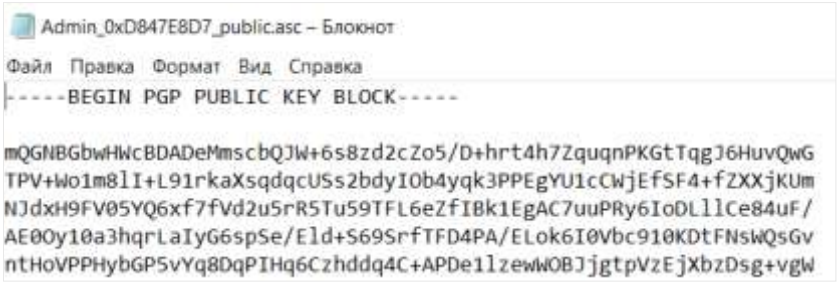


Рисунок 3.24 – Публічний ключ

Для імпорту публічного ключа іншого користувача вибираємо «Імпортувати…» та вибираємо відповідний файл ключа (рис. 3.25).

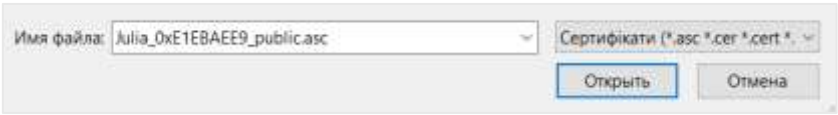


Рисунок 3.25 – Імпорт нового публічного ключа

Після вибору файлу відкривається вікно для посвідчення сертифіката (рис. 3.26).

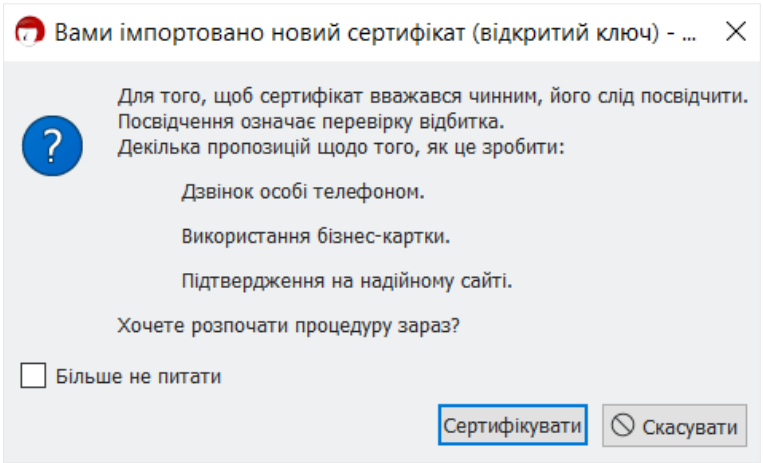


Рисунок 3.26 – Вікно посвідчення сертифіката

Ефективним способом переконатись у справжності ключа є відбиток сертифіката, що публікують разом із ключем. Якщо відбиток збігається з заявленим, посвідчуємо сертифікат (рис. 3.27).

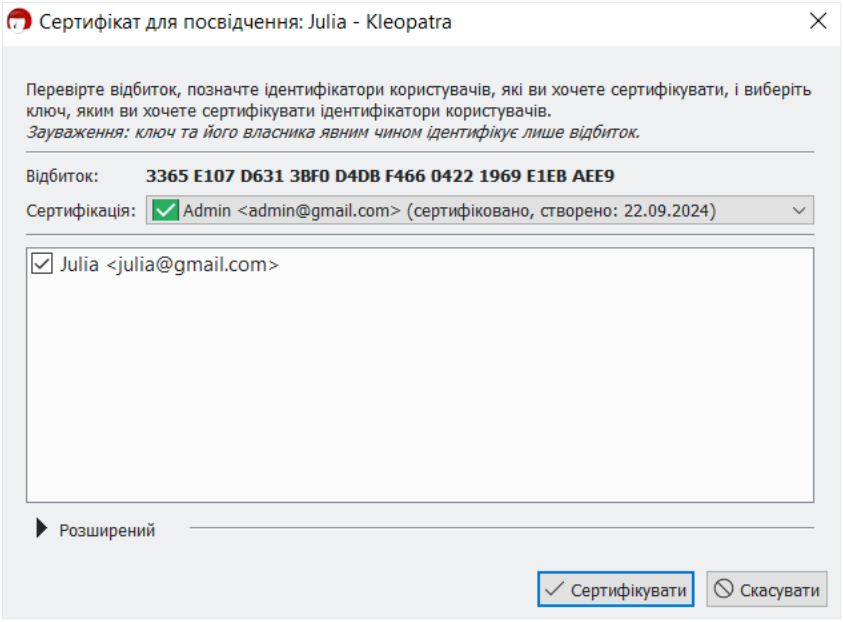


Рисунок 3.27 – Перевірка відбитка власника ключа

Після проведеної сертифікації отримаємо повідомлення про успішне її завершення (рис. 3.28).

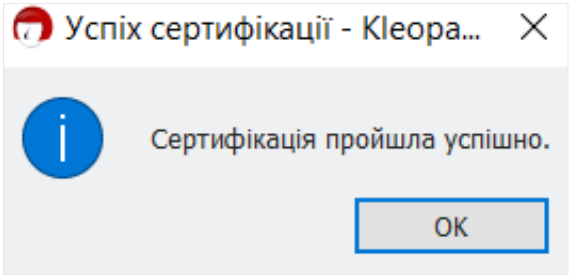


Рисунок 3.28 – Повідомлення про успішну сертифікацію

Для того, щоб підписати та зашифрувати файл, вибираємо відповідний пункт меню «Підписати/зашифрувати файли» (рис. 3.29).

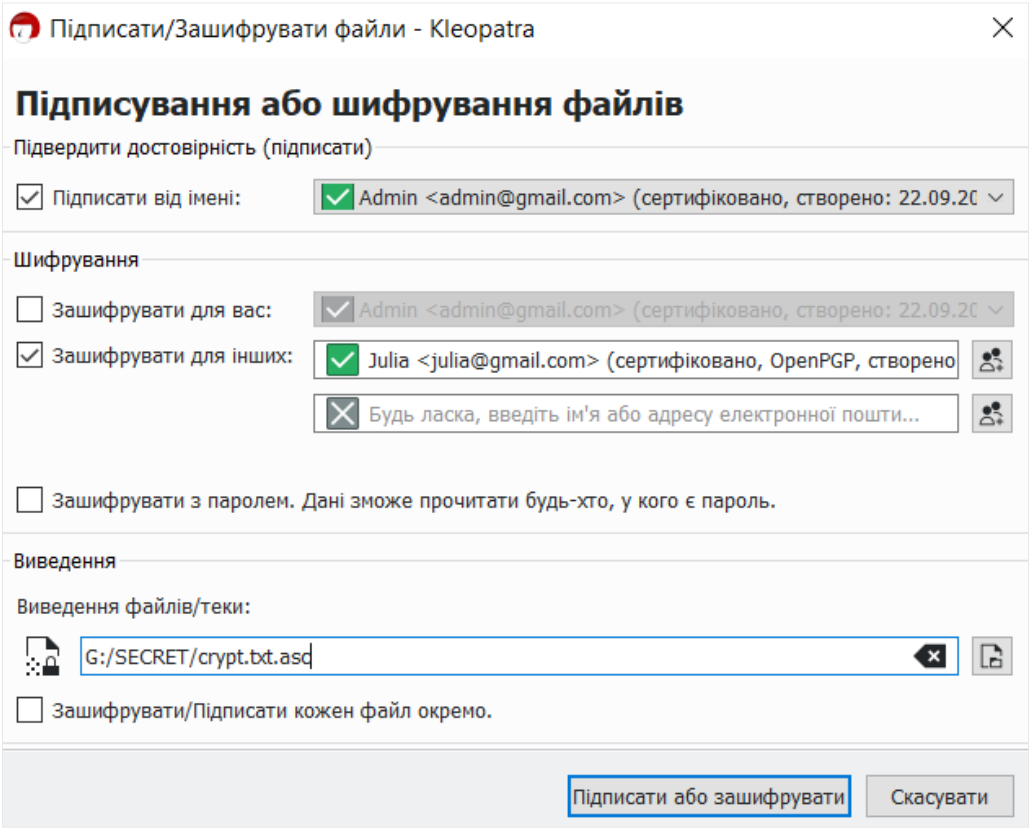


Рисунок 3.29 – Вікно для підписування / шифрування файлів

Після вибору файлу пропонується вибрати потрібні дії. Підпис дозволяє одержувачу переконатися у авторстві файлу. Для підпису використовуємо приватний ключ, після чого одержувач зможе перевірити автентичність за допомогою відкритого ключа відправника.

Для шифрування вибираємо відкритий ключ отримувача та натискаємо на кнопку «Підписати або зашифрувати». Після чого отримаємо повідомлення про успішне підписування і шифрування (рис. 3.30).

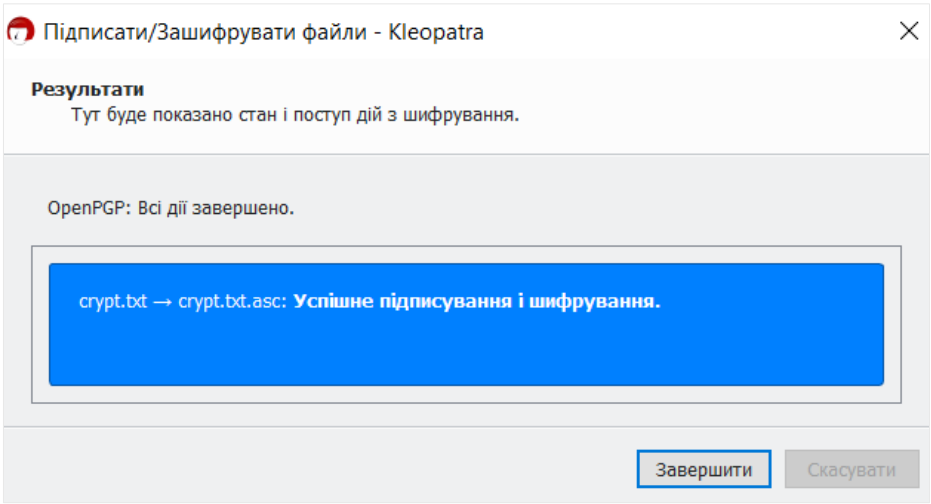


Рисунок 3.30 – Повідомлення про успішне підписування і шифрування

Для розшифрування та перевірки цифрового підпису відправника виберемо пункт меню «Розшифрувати/перевірити підпис». Після вибору файлу з’явиться вікно для введення пароля. Після правильного його введення файл буде розшифрований. Якщо файл був без цифрового підпису, то з’явиться повідомлення про те, що файл не має підпису (рис. 3.31).

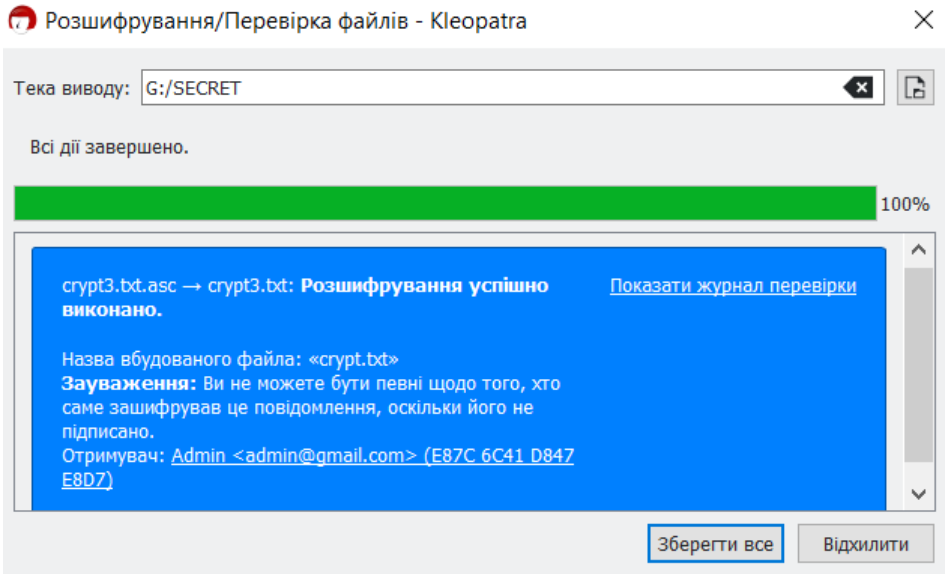


Рисунок 3.31 – Повідомлення про розшифрування файлу

Якщо файл має підпис, то з’явиться повідомлення про відправника, який підписав файл, дату створення підпису та його дійсність (рис. 3.32).

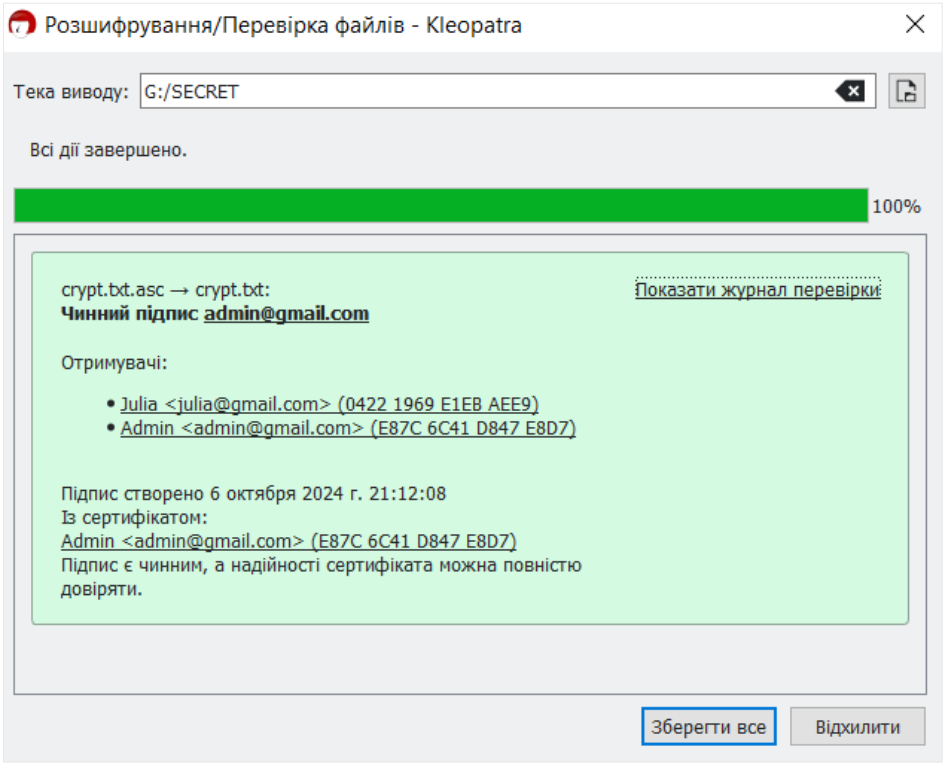


Рисунок 3.32 – Повідомлення про власника файлу

Натискаємо на кнопку «Зберегти все» для збереження розшифрованого файлу.

Для розшифрування файлу, зашифрованого публічним ключем, обов'язково потрібно мати відповідний приватний ключ, інакше спроба розшифрування зазнає невдачі (рис. 3.33).

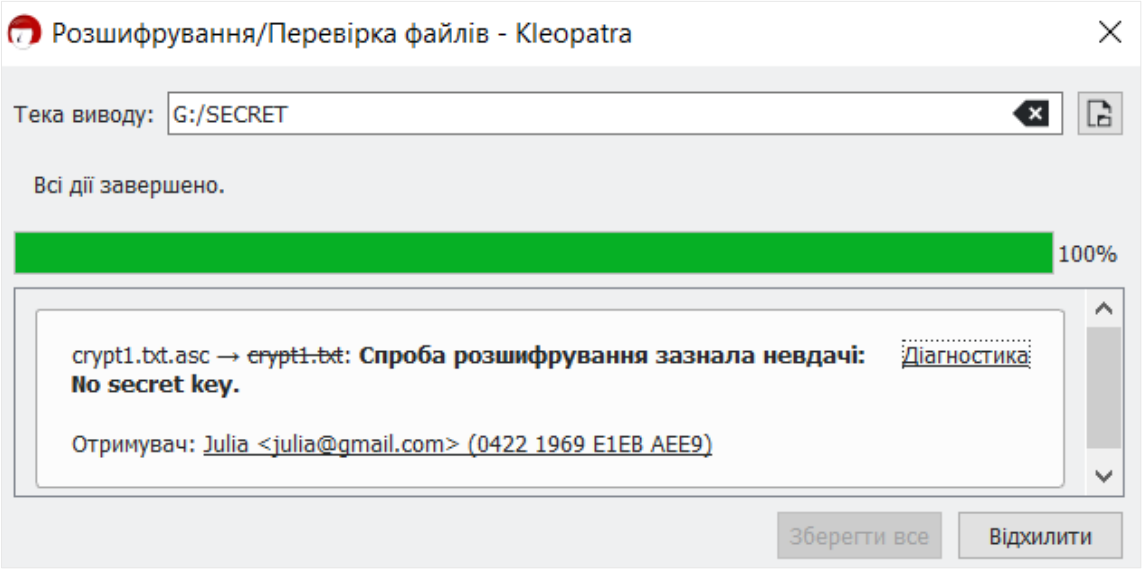


Рисунок 3.33 – Повідомлення про невдале розшифрування

Kleopatra дає можливість також створювати повідомлення з накладеним підписом без шифрування [37]. До повідомлення додається цифровий підпис, що дозволяє підтвердити автентичність і цілісність повідомлення. Текст при цьому залишається читабельним.

Для демонстрації даного прикладу накладання цифрового підпису відкриємо «Нотатник» та введемо повідомлення (рис. 3.34).

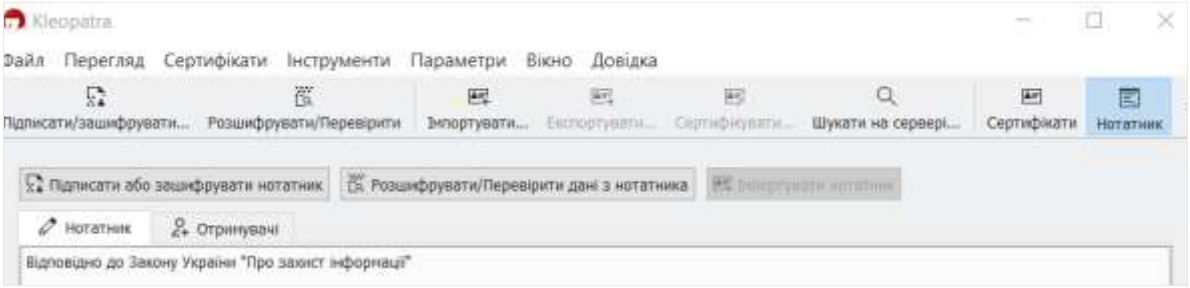


Рисунок 3.34 – Введення тексту в нотатнику

Обираємо від імені кого буде підписано документ (рис. 3.35).

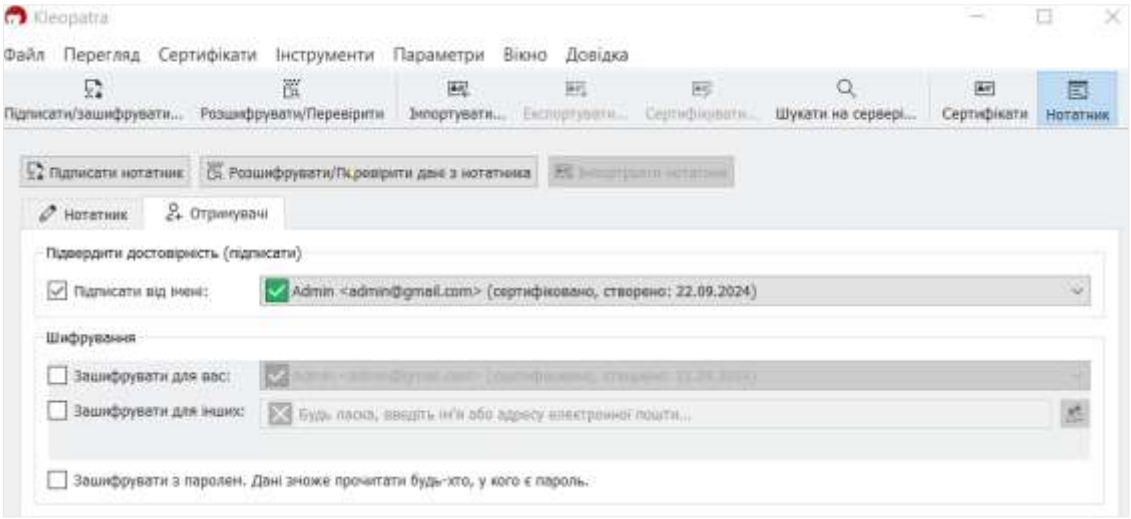


Рисунок 3.35 – Вибір власника цифрового підпису

Натискаємо на кнопку «Підписати нотатник» та вводимо пароль для розблокування приватного ключа. В результаті отримаємо текстове повідомлення, яке містить як оригінальний текст, так і цифровий підпис у спеціальному форматі (рис. 3.36).

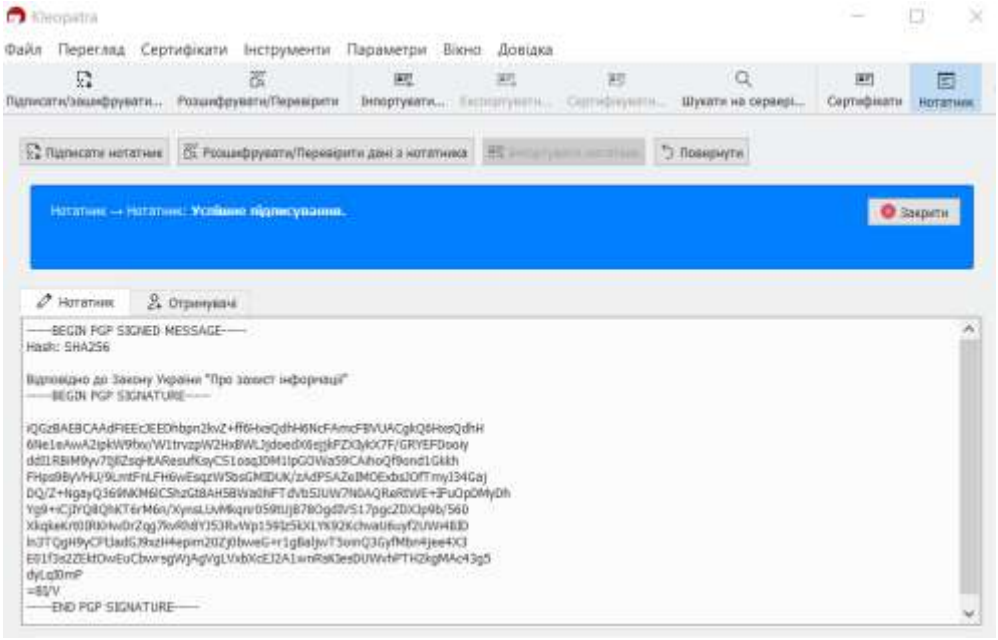


Рисунок 3.36 – Повідомлення із цифровим підписом

Будь-який суб’єкт, що отримав таке повідомлення, має можливість верифікувати підпис, використовуючи відкритий ключ відправника, це, в свою чергу, гарантує цілісність даних й ідентичність автора. Для перевірки вставимо скопійоване повідомлення в «Нотатник» та натиснемо на «Перевірити дані з нотатника». В результаті отримано повідомлення про те, що підпис є дійсним, а сертифікат відправника – надійним (рис. 3.37).

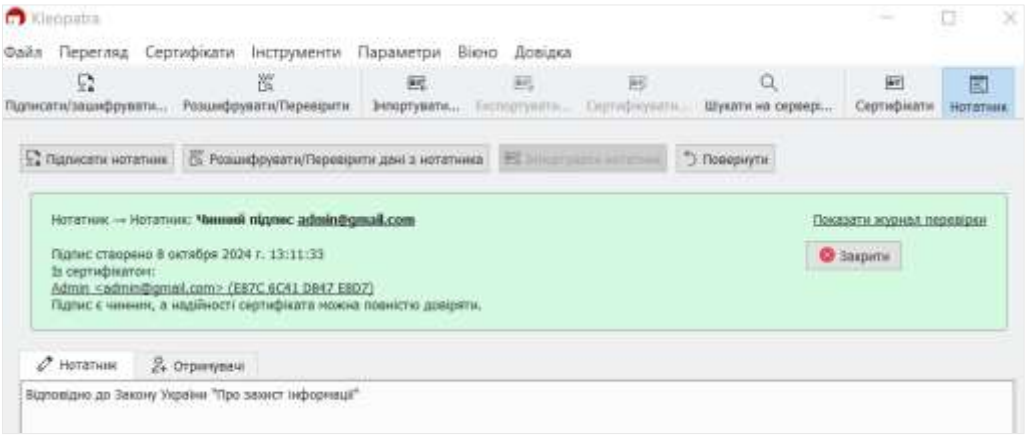


Рисунок 3.37 – Результат перевірки даних

Припустимо, що текст повідомлення було змінено (рис. 3.38).

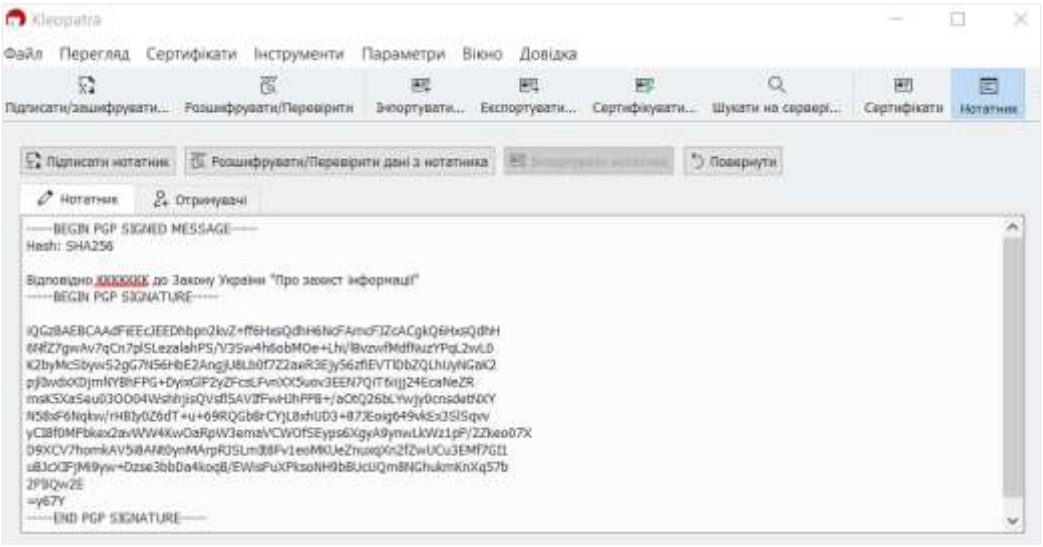


Рисунок 3.38 – Внесення змін у повідомлення

Після перевірки даних отримаємо повідомлення про те, що підпис цього повідомлення є недійсним (рис. 3.39).

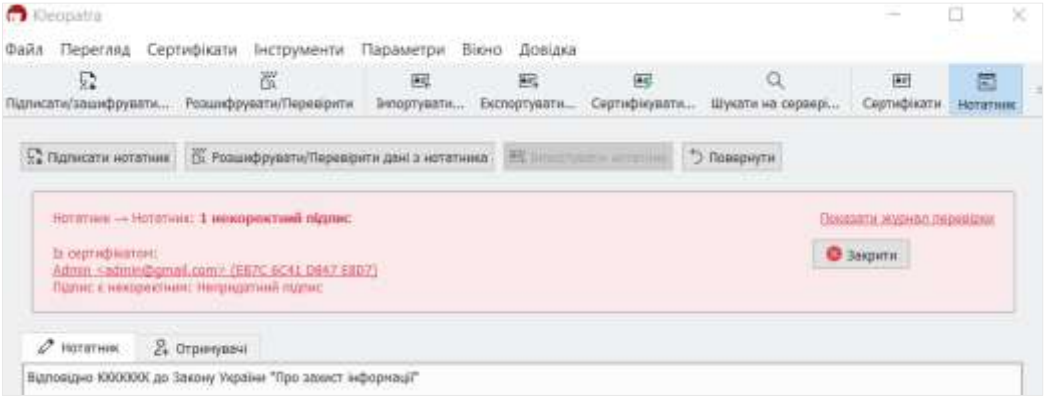


Рисунок 3.39 – Повідомлення про недійсний підпис

Отже, Kleopatra є невід'ємною частиною пакета Gpg4win і забезпечує зручний спосіб керування цифровими підписами, шифруванням та розшифруванням даних. Kleopatra легко інтегрується з такими програмами, як поштові клієнти, що дозволяє використовувати шифрування безпосередньо в робочому процесі та поширюється на умовах вільної ліцензії, що робить її доступною для всіх користувачів.

3.6 Контрольні питання

1. Яким чином генеруються відкритий та секретний ключі в криптоалгоритмі RSA?
2. Опишіть процедури зашифрування та розшифрування в алгоритмі RSA.
3. Від яких параметрів алгоритму RSA залежить його криптостійкість?
4. Опишіть можливі атаки на алгоритм RSA.
5. Які математичні основи лежать в основі протоколу Діффі-Хеллмана?
6. Які параметри потрібно визначити перед початком обміну ключами за протоколом Діффі-Хеллмана?
7. Як відбувається процес обміну ключами між двома сторонами за протоколом Діффі-Хеллмана?
8. Які математичні задачі лежать в основі стійкості протоколу Діффі-Хеллмана?
9. Які основні етапи генерації ключів в алгоритмі Ель-Гамаля?
10. Як здійснюється шифрування повідомлення за допомогою алгоритму Ель-Гамаля?
11. Який механізм використовується для розшифрування повідомлення в алгоритмі Ель-Гамаля?
12. На якій математичній задачі базується криптостійкість алгоритму Ель-Гамаля?
13. Як впливає розмір простого числа p на безпеку алгоритму Ель-Гамаля?
14. Які є основні вразливості алгоритму Ель-Гамаля?
15. Яка роль простих чисел у процесі шифрування та цифрового підписування на основі криптоалгоритму RSA?
16. Які основні етапи генерації цифрового підпису в алгоритмі Фіата-Шаміра?
17. Як працює процес перевірки підпису в протоколі Фіата-Шаміра?
18. Як генерується пара ключів у схемі цифрового підпису Ель-Гамаля?
19. Яким чином здійснюється підпис і верифікація повідомлення в алгоритмі Ель-Гамаля?
20. Що таке дискретне логарифмування і яку роль воно відіграє в алгоритмі Ель-Гамаля?

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Основи криптографічного захисту інформації : підручник / Гулак Г М., Мухачов В. А., Хорошко В. О., Яремчук Ю. Є. Вінниця : ВНТУ, 2011. 199 с.

2. Paar C., Pelzl J. Understanding Cryptography : A Textbook for Students and Practitioners. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. 390 p. URL: https://doi.org/10.1007/978-3-642-04101-3 (дата звернення: 09.07.2024)

3. Гапак О. М. Криптоаналіз. Криптографічні протоколи : навч. посіб. для студ. спец. 123 «Комп'ютерна інженерія». Ужгород : ПП «АУТДОР-ШАРК», 2021. 93 с.

4. Щур Н. О., Покотило О. А. Основи криптології : навч. посіб. Житомир : Державний університет «Житомирська політехніка», 2021. 120 с.

5. Jones M. N., Mewhort D. J. K. Case-sensitive letter and bigram frequency counts from large-scale English corpora. *Behavior Research Methods, Instruments, & Computers* 36, 2004. P. 388–396. URL: https://doi.org/10.3758/BF03195586 (дата звернення: 11.07.2024).

6. Cryptography Online. A. A. Mardon et al. Academia.edu - Find Research Papers, Topics, Researchers. URL: https://www.academia.edu/53297148/ Cryptography_Online (дата звернення: 16.07.2024).

7. Євсеєв С. П., Мілов О. В., Король О. Г. Кібербезпека. Лабораторний практикум з основ криптографічного захисту : навч. посіб. Львів : «Новий Світ- 2000», 2021. 241 с.

8. Menezes A. J., Oorschot P. C. v., Vanstone S. A. Handbook of Applied Cryptography (Discrete Mathematics and Its Applications). 5th ed. CRC Press, Inc., 1997. 780 p.

9. Banoth R., Regar R. Classical and Modern Cryptography for Beginners. Switzerland : Springer International Publishing AG, 2023. 215 p.

10. CrypTool 2. CrypTool Portal. URL: https://www.cryptool.org/ en/ct2/ (дата звернення: 22.07.2024).

11. What is Cryptography? Definition, Importance, Types | Fortinet. Fortinet. URL: https://www.fortinet.com/resources/ cyberglossary/what-is-cryptography (дата звернення: 25.07.2024).

12. Letter Frequencies in English. Welcome. URL: https://mathcenter.oxford.emory.edu/site/math125/englishLetterFreqs/ (дата звернення: 29.07.2024).

13. The Vigenère Cipher Encryption and Decryption. Article - Managing your personal web ... URL: https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html (дата звернення: 05.08.2024).

14. Hassan A. ANALYSIS AND MODIFICATION OF VIGENERE CIPHER. Journal of Mathematical Sciences & Computational Mathematics, 2024. Vol. 5, No. 4. P. 502–510.

15. Forcinito M. A., Bruen A. A., McQuillan J. M. Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century. 2nd Edition. Wiley & Sons, Incorporated, John, 2021. 656 p.

16. The Black Chamber. URL: https://simonsingh.net/ The_Black_Chamber/ index.html (дата звернення: 11.08.2024).

17. Data Encryption Standard. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/files/pubs/fips/46-3/final/docs/fips46-3.pdf (дата звернення: 16.08.2024).

18. Oppliger R. Cryptography 101: From Theory to Practice. Artech House, 2021. 679 p.

19. National Institute of Standards and Technology (2001) Advanced Encryption Standard (AES). (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) NIST FIPS 197-upd1, updated May 9, 2023. URL: https://doi.org/10.6028/NIST.FIPS.197-upd1 (дата звернення: 19.08.2024).

20. National Institute of Standards and Technology. AES Development, 2022. URL: https://csrc.nist.gov/projects/aes (дата звернення: 24.08.2024).

21. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. [Текст]. Введ. 01-07-2015. К. : Мінекономрозвитку України, 2015.

22. Горбенко І. Д. та ін. Симетричний блоковий шифр «Калина» – новий національний стандарт України. *Радіотехніка*. 2015. Вип. 181. С. 5–22. URL: http://nbuv.gov.ua/UJRN/rvmnts_2015_181_3 (дата звернення: 03.09.2024).

23. Совин Я. Р., Хома В. В., Наконечний Ю. М, Стахів М. Ю. Ефективна імплементація та порівняння швидкодії шифрів «КАЛИНА» та ГОСТ 28147-89 за використання векторних розширень SSE, AVX та AVX-512. *Захист інформації*. Т. 21, № 4. С. 207–223.

24. Козіна Г. Л. Криптографія від історії до сучасних стандартів : навч. посіб. Запоріжжя : НУ «Запорізька політехніка», 2020. 192 с.

25. A New Encryption Standard of Ukraine: The Kalyna Block Cipher / R. Oliynykov et al. IACR Cryptology ePrint Archive. URL: https://eprint.iacr.org/ 2015/650 (дата звернення: 12.09.2024).

26. Горбенко Ю. І., Мордвінов Р. І., Кузнецов О. О. Розробка математичних та програмних моделей перспективного алгоритму шифрування для перевірки правильності реалізації. *Eastern-European Journal of Enterprise Technologies.* 2014. № 5/9 (71). С. 39–45.

27. ДСТУ 8845:2019. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення. Чинний від 2019–10–01. Київ : Держстандарт України.

28. Корченко А. О. та ін..Стандартизація систем, комплексів та засобів криптографічного захисту інформації для застосування у постквантовому середовищі. *Ukrainian Information Security Research Journal.* 2023. Т. 22, № 4. С. 227–254. URL: https://doi.org/10.18372/2410-7840.22.15257 (дата звернення: 20.09.2024).

29. Кузнецов О. О., Горбенко І. Д., Горбенко Ю. І., Олексійчук А. М., Тимченко В. А. Математична структура потокового шифру Струмок. *Радіотехніка.* 2018. 2(193). С. 17–27. URL: https://doi.org/10.30837/rt.2018.2.193.02 (дата звернення: 23.09.2024).

30. Rivest R. L., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*. 21 (2). P. 120–126. URL: https://dl.acm.org/doi/10.1145/359340.359342 (дата звернення: 25.09.2024).

31. Кібер-безпека: основи кодування та криптографії. / Євсеєв С. П., Мілов О. В., Остапов С. Е., Сєвєрінов О. В. Харків : Вид. «Новий Світ-2000», 2023. 657 с.

32. ElGamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *Advances in Cryptology: Proceedings of CRYPTO 84.* Springer Verlag, 1985. P. 1–18.

33. Яремчук Ю. Є. Дослідження статистичної безпеки методів цифрового підписування на основі рекурентних послідовностей. *Реєстрація, зберігання і обробка даних*. 2014. Т. 16, № 2. С. 74–86.

34. Яремчук Ю. Є. Метод цифрового підписування на основі рекурентних послідовностей. *Інформаційна безпека*, 2013. № 1 (9) С. 165–175.

35. Галкін О. В., Шкільняк О. С. Основи криптології : навч. посіб. – Київ, 2023. 119 с.

36. Gpg4win – Secure email and file encryption with GnuPG for Windows. Gpg4win – Secure email and file encryption with GnuPG for Windows. URL: https://www.gpg4win.org/index.html (дата звернення: 29.09.2024).

37. The Kleopatra Handbook. KDE Documentation. URL: https://docs.kde.org/stable5/en/kleopatra/kleopatra/index.html (дата звернення: 02.10.2024).

ДОДАТКИ

ДОДАТОК А

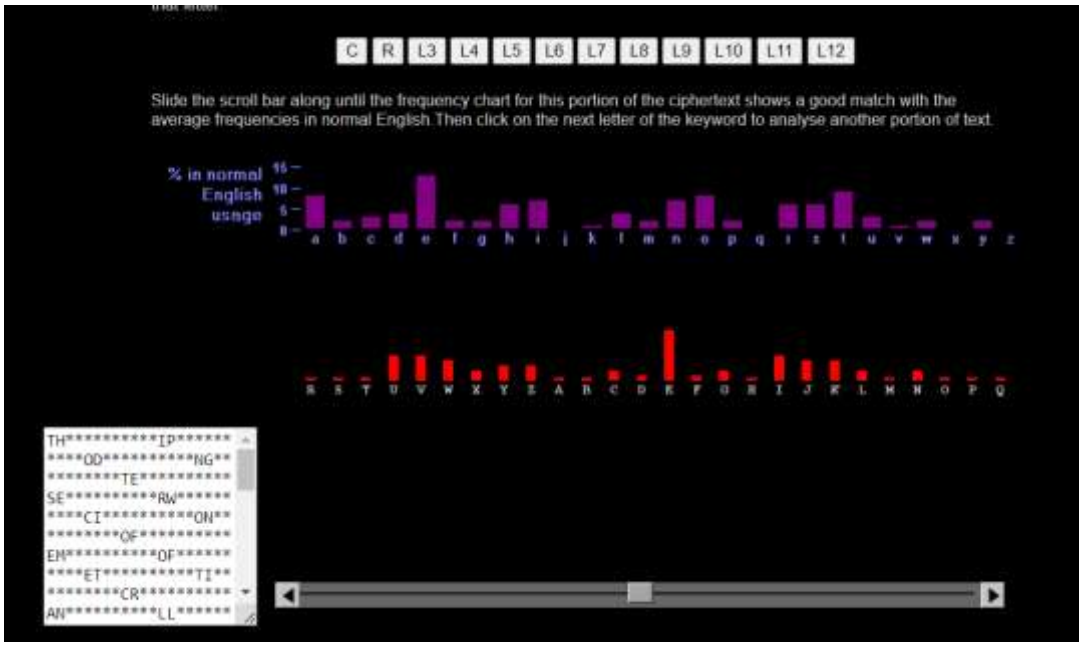
Таблиця логарифмів ймовірностей біграм англійського тексту

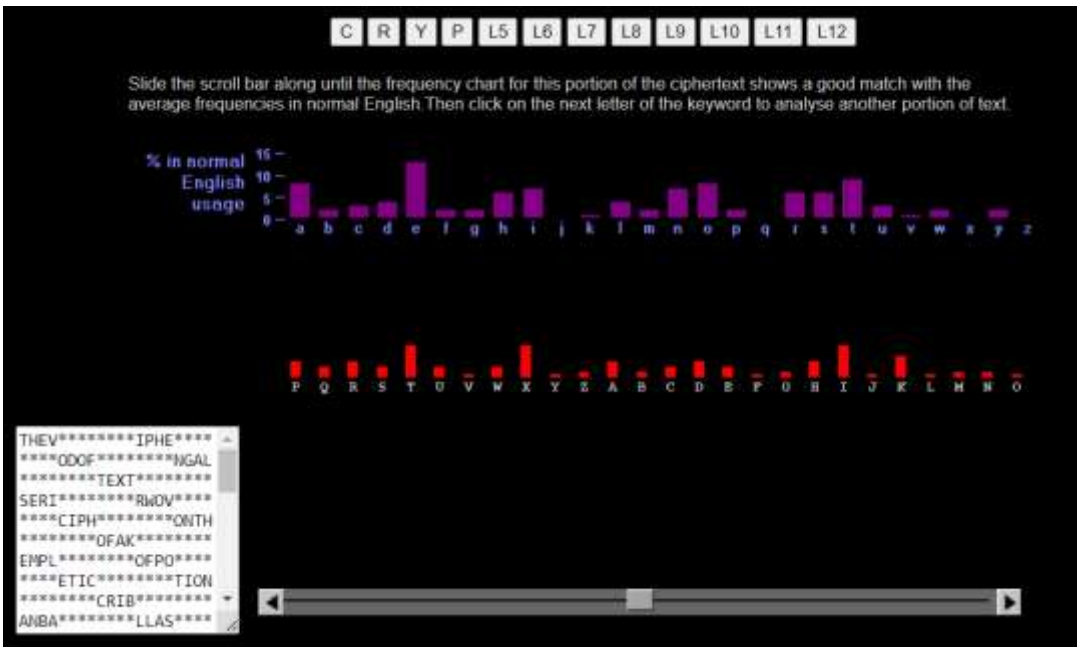
	A	B	C	D	E	F	G	H	I	J	K	L	M
A	7,38	11,5	12,2	12,1	9,25	10,6	11,6	9,03	12,4	9,03	11	13,1	11,9
B	11,4	9	6,57	6,5	12,4	3,85	3,56	5,33	10,9	8,14	3,56	11,5	6,93
C	12,4	4,08	10,4	5,76	12,7	3	0,69	12,5	11,7	1,1	11,4	11,1	4,89
D	11,6	7,33	7,73	10,1	12,8	7,46	9,83	7,55	12,2	7,54	5,46	9,76	9,44
E	12,8	10,2	12,4	13,3	12,2	11,2	11	9,36	11,3	7,58	9,97	12,4	12
F	11,1	4,32	4,78	3,5	11,5	11,3	6,02	3,64	11,9	4,23	5,42	10,1	6,27
G	11,4	5,36	3,58	6,81	12,2	6,2	9,57	11,7	11	3,09	4,68	9,99	7,66
H	13,1	8,2	6,35	7,84	14,1	6,6	5,24	6,21	12,6	3,5	5,91	9	8,58
I	11,9	10,4	12,8	12,4	12,1	11,1	11,8	6,54	7,1	7,21	10,4	12,5	11,8
J	8,52	1,61	3,43	3,56	9,39	2,08	.	1,1	7,24	2,3	3,76	2,4	3,93
K	9,21	6,78	4,55	6,64	11,9	6,73	6,35	7,68	11	3,85	6,42	9,22	6,53
L	12,5	8,6	8,44	11,9	12,9	10	7,95	7,18	12,7	4,22	9,54	12,7	9,68
M	12,3	10,7	5,82	5,65	12,8	7,59	4,8	5,64	12	3,56	4,3	7,21	10,8
N	12	8,42	12,2	13,3	12,8	10,2	13,2	9,07	12,1	8,4	10,5	10,1	9,81
O	10,5	10,6	11,2	11,3	9,82	13	10,6	9,43	10,8	8,47	10,6	12,1	12,6
P	12	6,81	5,74	5,81	12,3	6,47	7,8	10,4	11	4,91	6,47	11,8	8,67
Q	3,3	1,95	.	0	0,69	0,69	0	.	6,64	.	.	1,1	.
R	12,8	9,63	11,2	11,6	13,7	9,61	11	8,83	12,7	5,58	11,4	10,8	11,3
S	12,1	8,59	11,1	9,1	12,9	8,62	6,66	12	12,5	4,16	10,2	10,4	10,1
T	12,5	8,43	9,81	7,32	13,3	8,08	7,71	14,2	13,2	4,17	6,14	10,7	9,83
U	10,9	10,8	11,3	10,9	11,2	9,19	11,1	6,29	10,8	5,98	8,08	11,9	11
V	10,8	0	6,05	4,2	12,9	0	4,52	1,39	12	0,69	3,18	5,15	2,4
W	12,1	6,89	6,09	7,88	11,8	6,22	3,71	12	12,1	.	7,07	8,85	7,31
X	9,11	3,78	9,11	0	9,61	5,59	1,1	7,76	9,11	.	.	5,06	4,39
Y	9,16	8,22	8,48	7,77	11,4	6,35	6,34	6,24	9,92	3,09	5,96	9,07	9,49
Z	9,41	5,86	3,69	4,73	10,1	2,56	5	6,32	8,93	0,69	5,29	6,95	6
	A	B	C	D	E	F	G	H	I	J	K	L	M

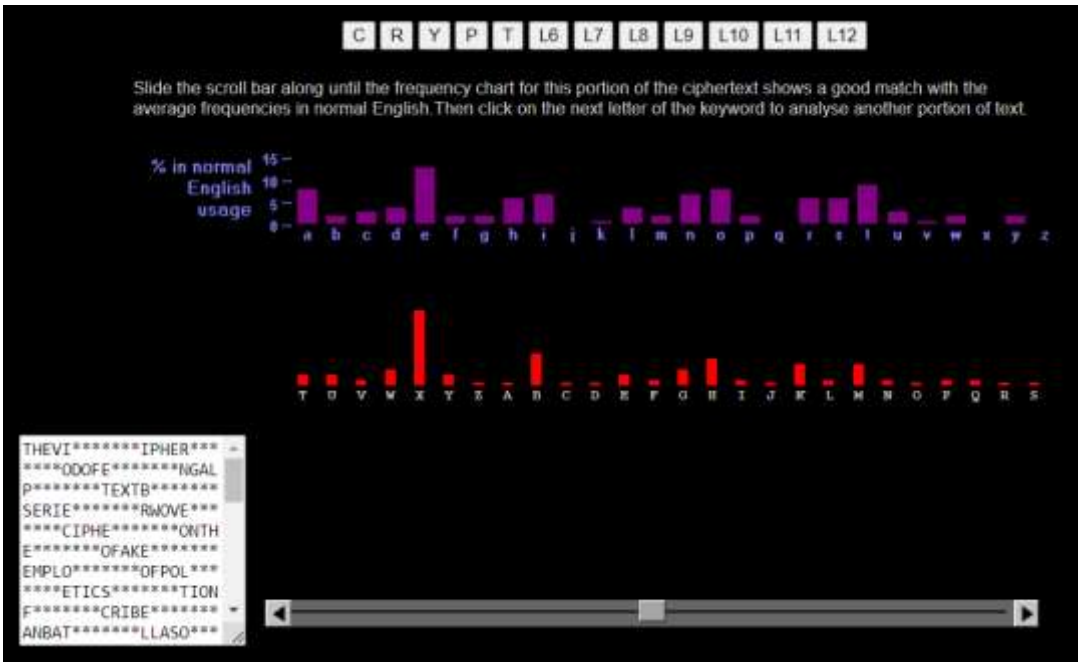
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	13,8	7,79	11,4	7,95	13,3	13	13,5	11,1	11,6	10,5	9,64	12	9,23
B	5,61	11,5	5,25	.	10,6	10,2	8,52	11,3	7,05	6,57	.	11,4	2,71
C	4,9	12,8	3,76	7,87	11,2	9,47	12,1	11,2	1,95	3	0,69	9,93	5,82
D	9,16	11,4	6,09	6,83	10,7	11	7,3	11	9,31	8,73	0	9,98	5,09
E	13,4	10,6	11,4	9,52	13,9	13,4	12,3	9,23	11,6	11,5	11,3	11,4	8,43
F	5,31	12,4	3,26	.	11,5	8,16	10,8	10,5	3,81	4,91	1,39	8,18	2,56
G	10,4	11	5,28	1,95	11,3	10,3	9,35	10,6	2,94	5,96	2,08	9,12	3,64
H	9,71	12,4	5,91	5,12	10,6	9,32	11,2	10,2	5,18	8,07	.	9,54	2,64
I	14	12,7	10,5	8,09	11,9	13,1	13,1	8,71	11,7	6,71	9,25	6,69	10,2
J	3,37	10	1,61	.	1,39	2,3	2,56	10,1	0,69	.	.	1,39	1,79
K	9,8	8,42	6,3	.	8,04	10,5	6,95	7,46	5,42	7,05	0	8,7	2,56
L	7,78	12	9,5	3,43	8,53	11,3	10,9	10,9	9,58	8,49	3,4	12,1	6,25
M	7,95	12	11,8	1,39	5,81	10,6	5,84	10,7	4,43	4,88	.	9,84	2,89
N	10,9	12,1	7,93	7,36	8,38	12,4	13,1	10,7	10,3	8,3	7,55	11,1	7,75
O	13,6	11,7	11,7	6,27	13,4	11,9	12,2	12,9	11,5	12	8,62	9,85	8,3
P	5,6	12	11,1	2,08	12,2	10,2	10,4	10,9	1,1	5,85	0	8,97	3,58
Q	0	0	.	1,1	.	1,1	1,95	10,9	1,79	0,69	.	.	.
R	11,5	12,8	10,1	6,18	11	12,5	12,2	11	11	8,79	5,47	11,6	6,92
S	9,18	12	11,3	8,25	8,72	12,2	13,2	11,7	6,44	9,15	0	9,75	5,97
T	8,95	13,2	7,33	1,95	12,2	12,2	11,4	11,4	6,49	10,5	2,83	11,5	8,49
U	12,1	8,53	11,2	5,73	12,3	12,3	12,4	5,42	7,31	6,43	7,56	8,94	7,43
V	5,34	10,4	.	.	6,23	6,43	3,71	6,76	4,74	.	.	8,11	1,1
W	10,7	11,7	5,53	1,95	9,53	10	7,87	4,69	0,69	2,94	0	8,43	2,2
X	2,94	7,47	10,3	4,76	1,39	4,55	9,75	7,96	4,62	6,11	5,06	6	0
Y	9,11	10,6	8,63	1,61	8,97	10,7	8,86	6,57	5,64	8,04	4,42	3,43	6,56
Z	4,67	8,13	4,98	3,14	4,75	4,98	4,84	6,85	4,84	5,21	.	7,23	8,1
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

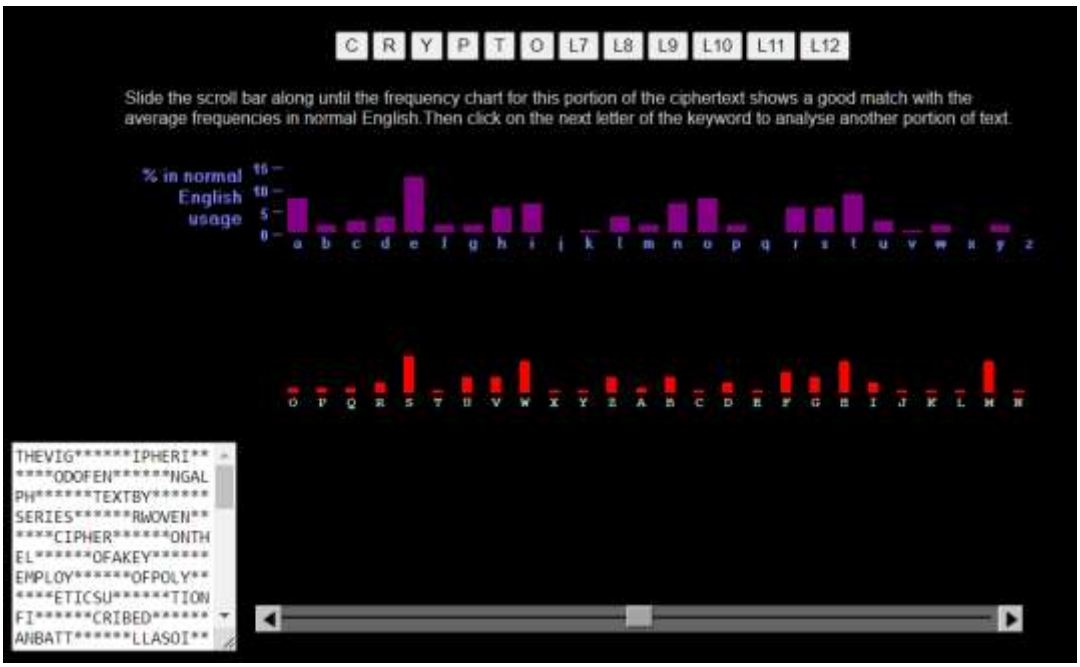
ДОДАТОК Б

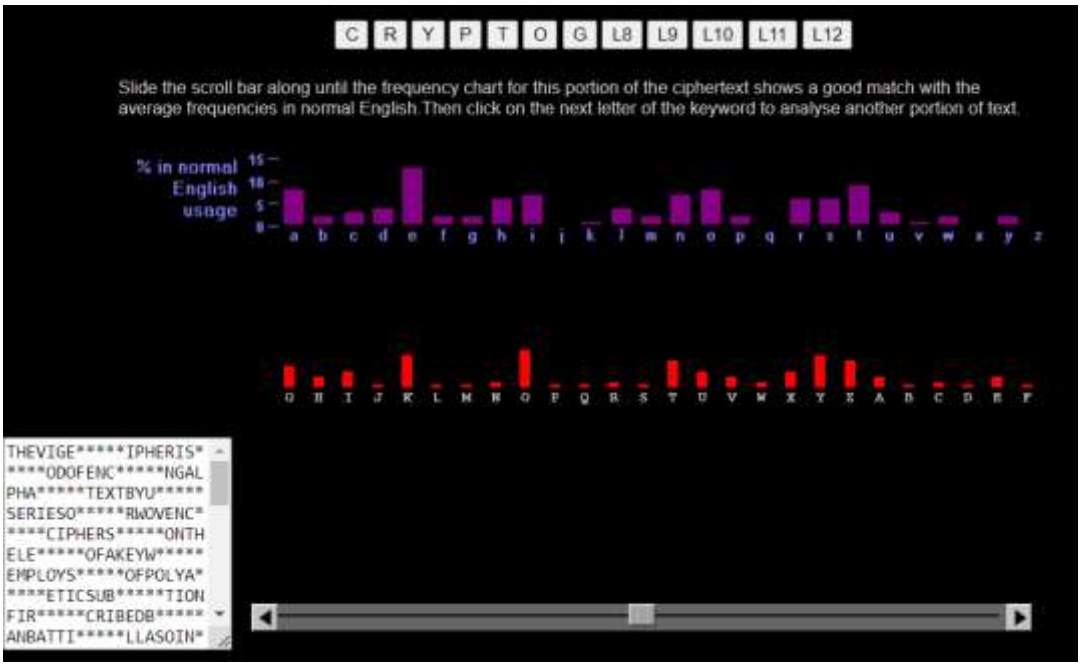
Визначення ключової послідовності

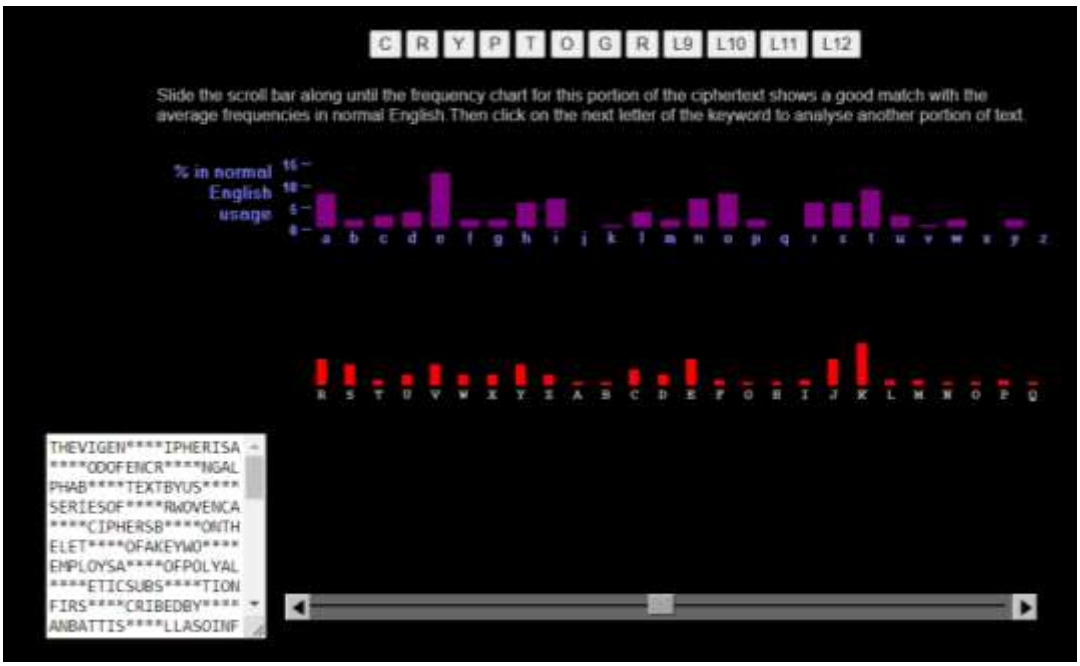


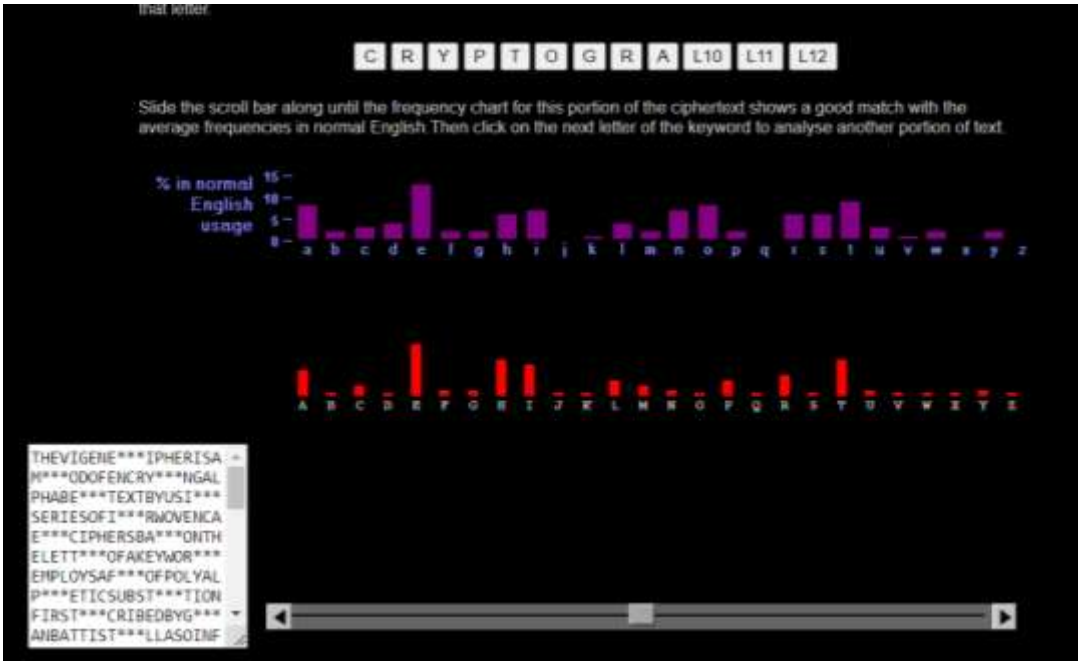

C R Y P L5 L6 L7 L8 L9 L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
15
10
5
0
a b c d e f g h i j k l m n o p q r s t u v w x y z
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
THEV********IPHE****
****ODOF********NGAL
********TEXT********
SERI********RWOV****
****CIPH********ONTH
********OFAK********
EMPL********OFPO****
****ETIC********TION
********CRIB********
ANBA********LLAS****


C R Y P T L6 L7 L8 L9 L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
15
10
5
0
a b c d e f g h i j k l m n o p q r s t u v w x y z
T U V W X Y Z A B C D E F G H I J K L M N O P Q R S
THEVI*******IPHER***
****ODOFE*******NGAL
P********TEXTB*******
SERIE*******RWOVE***
****CIPHE*******ONTH
E********OFAKE*******
EMPLO*******OFPOL***
****ETICS*******TION
F********CRIBE*******
ANBAT*******LLASO***

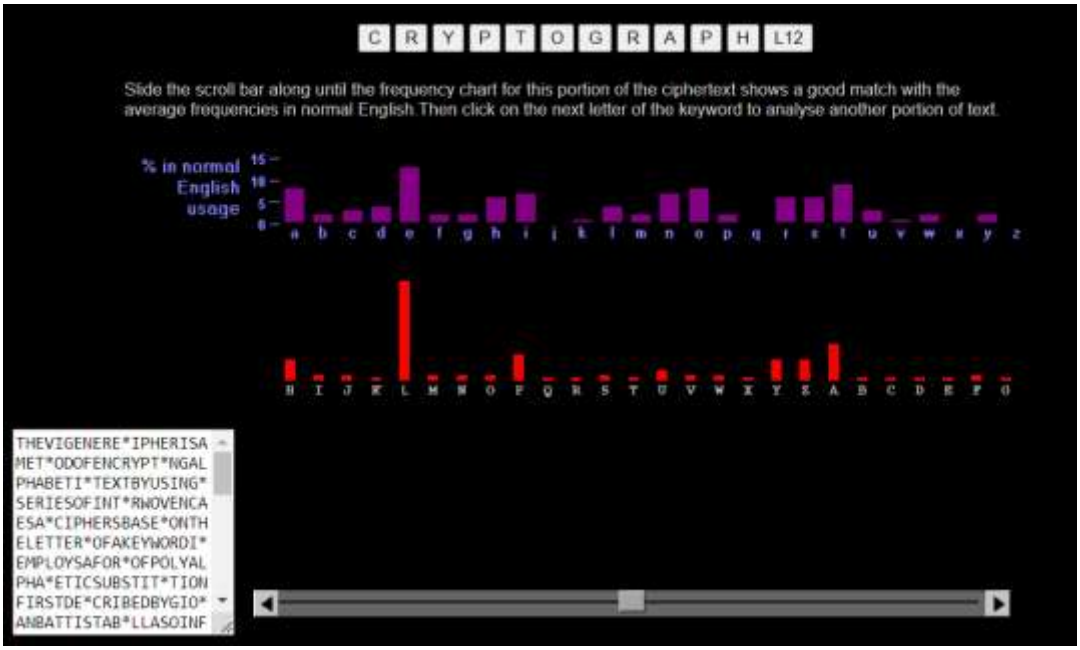

C R Y P T O L7 L8 L9 L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
THEVIG******IPHERI**
****ODOFEN******NGAL
PH******TEXTBY******
SERIES******RWOVEN**
****CIPHER******ONTH
EL******OFAKEY******
EMPLOY******OFPOLY**
****ETICSU******TION
FI******CRIBED******
ANBATT******LLASOI**


C R Y P T O G L8 L9 L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
THEVIGE*****IPHERIS*
****ODOFENC*****NGAL
PHA*****TEXTBYU*****
SERIESO*****RWOVENC*
****CIPHERS*****ONTH
ELE*****OFAKEYW*****
EMPLOYS*****OFPOLYA*
****ETICSUB*****TION
FIR*****CRIBEDB*****
ANBATTI*****LLASOIN*


C R Y P T O G R L9 L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
THEVIGEN****IPHERISA
****ODOFENCR****NGAL
PHAB****TEXTBYUS****
SERIESOF****RWOVENCA
****CIPHERSB****ONTH
ELET****OFAKEYWO****
EMPLOYSA****OFPOLYAL
****ETICSUBS****TION
FIRS****CRIBEDBY****
ANBATTIS****LLASOINF


C R Y P T O G R A L10 L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
THEVIGENE***IPHERISA
M***ODOFENCRY***NGAL
PHABE***TEXTBYUSI***
SERIESOFI***RWOVENCA
E***CIPHERSBA***ONTH
ELETT***OFAKEYWOR***
EMPLOYSAF***OFPOLYAL
P***ETICSUBST***TION
FIRST***CRIBEDBYG***
ANBATTIST***LLASOINF


C R Y P T O G R A P L11 L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
15
10
5
0
a b c d e f g h i j k l m n o p q r s t u v w x y z
P Q R S T U V W X Y Z A B C D E F G H I J K L M N O
THEVIGENER**IPHERISA
ME**ODOFENCRYP**NGAL
PHABET**TEXTBYUSIN**
SERIESOFIN**RWOVENCA
ES**CIPHERSBAS**ONTH
ELETTE**OFAKEYWORD**
EMPLOYSAFO**OFPOLYAL
PH**ETICSUBSTI**TION
FIRSTD**CRIBEDBYGI**
ANBATTISTA**LLASOINF


C R Y P T O G R A P H L12
Slide the scroll bar along until the frequency chart for this portion of the ciphertext shows a good match with the average frequencies in normal English.Then click on the next letter of the keyword to analyse another portion of text.
% in normal English usage
15
10
5
0
a b c d e f g h i j k l m n o p q r s t u v w x y z
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
THEVIGENERE*IPHERISA
MET*ODOFENCRYPT*NGAL
PHABETI*TEXTBYUSING*
SERIESOFINT*RWOVENCA
ESA*CIPHERSBASE*ONTH
ELETTER*OFAKEYWORDI*
EMPLOYSAFOR*OFPOLYAL
PHA*ETICSUBSTIT*TION
FIRSTDE*CRIBEDBYGIO*
ANBATTISTAB*LLASOINF

ДОДАТОК В

Таблиця перетворення S_i -блоків

		Номер стовпця																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Номер рядка	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S1
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S2
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S3
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S4
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S5
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S6
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S7
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S8
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

ДОДАТОК Г

Таблиця InvS-Box оберненої заміни

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	52	9	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	8	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	0	8C	BC	D3	0A	F7	E4	58	5	B8	B3	45	6
7	D0	2C	1E	8F	CA	3F	0F	2	C1	AF	BD	3	1	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	7	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	4	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

ДОДАТОК Д

Таблиці підстановок в СБШ «Калина»

Підстановка π0

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	A8	43	5F	6	6B	75	6C	59	71	DF	87	95	17	F0	D8	9
1	6D	F3	1D	CB	C9	4D	2C	AF	79	E0	97	FD	6F	4B	45	39
2	3E	DD	A3	4F	B4	B6	9A	0E	1F	BF	15	E1	49	D2	93	C6
3	92	72	9E	61	D1	63	FA	EE	F4	19	D5	AD	58	A4	BB	A1
4	DC	F2	83	37	42	E4	7A	32	9C	CC	AB	4A	8F	6E	4	27
5	2E	E7	E2	5A	96	16	23	2B	C2	65	66	0F	BC	A9	47	41
6	34	48	FC	B7	6A	88	A5	53	86	F9	5B	DB	38	7B	C3	1E
7	22	33	24	28	36	C7	B2	3B	8E	77	BA	F5	14	9F	8	55
8	9B	4C	FE	60	5C	DA	18	46	CD	7D	21	B0	3F	1B	89	FF
9	EB	84	69	3A	9D	D7	D3	70	67	40	B5	DE	5D	30	91	B1
A	78	11	1	E5	0	68	98	A0	C5	2	A6	74	2D	0B	A2	76
B	B3	BE	CE	BD	AE	E9	8A	31	1C	EC	F1	99	94	AA	F6	26
C	2F	EF	E8	8C	35	3	D4	7F	FB	5	C1	5E	90	20	3D	82
D	F7	EA	0A	0D	7E	F8	50	1A	C4	7	57	B8	3C	62	E3	C8
E	AC	52	64	10	D0	D9	13	0C	12	29	51	B9	CF	D6	73	8D
F	81	54	C0	ED	4E	44	A7	2A	85	25	E6	CA	7C	8B	56	80

Підстановка π1

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	CE	BB	EB	92	EA	CB	13	C1	E9	3A	D6	B2	D2	90	17	F8
1	42	15	56	B4	65	1C	88	43	C5	5C	36	BA	F5	57	67	8D
2	31	F6	64	58	9E	F4	22	AA	75	0F	2	B1	DF	6D	73	4D
3	7C	26	2E	F7	8	5D	44	3E	9F	14	C8	AE	54	10	D8	BC
4	1A	6B	69	F3	BD	33	AB	FA	D1	9B	68	4E	16	95	91	EE
5	4C	63	8E	5B	CC	3C	19	A1	81	49	7B	D9	6F	37	60	CA
6	E7	2B	48	FD	96	45	FC	41	12	0D	79	E5	89	8C	E3	20
7	30	DC	B7	6C	4A	B5	3F	97	D4	62	2D	6	A4	A5	83	5F
8	2A	DA	C9	0	7E	A2	55	BF	11	D5	9C	CF	0E	0A	3D	51
9	7D	93	1B	FE	C4	47	9	86	0B	8F	9D	6A	7	B9	B0	98
A	18	32	71	4B	EF	3B	70	A0	E4	40	FF	C3	A9	E6	78	F9
B	8B	46	80	1E	38	E1	B8	A8	E0	0C	23	76	1D	25	24	5
C	F1	6E	94	28	9A	84	E8	A3	4F	77	D3	85	E2	52	F2	82
D	50	7A	2F	74	53	B3	61	AF	39	35	DE	CD	1F	99	AC	AD
E	72	2C	DD	D0	87	BE	5E	A6	EC	4	C6	3	34	FB	DB	59
F	B6	C2	1	F0	5A	ED	A7	66	21	7F	8A	27	C7	C0	29	D7

Підстановка π2

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	93	D9	9A	B5	98	22	45	FC	BA	6A	DF	2	9F	DC	51	59
1	4A	17	2B	C2	94	F4	BB	A3	62	E4	71	D4	CD	70	16	E1
2	49	3C	C0	D8	5C	9B	AD	85	53	A1	7A	C8	2D	E0	D1	72
3	A6	2C	C4	E3	76	78	B7	B4	9	3B	0E	41	4C	DE	B2	90
4	25	A5	D7	3	11	0	C3	2E	92	EF	4E	12	9D	7D	CB	35
5	10	D5	4F	9E	4D	A9	55	C6	D0	7B	18	97	D3	36	E6	48
6	56	81	8F	77	CC	9C	B9	E2	AC	B8	2F	15	A4	7C	DA	38
7	1E	0B	5	D6	14	6E	6C	7E	66	FD	B1	E5	60	AF	5E	33
8	87	C9	F0	5D	6D	3F	88	8D	C7	F7	1D	E9	EC	ED	80	29
9	27	CF	99	A8	50	0F	37	24	28	30	95	D2	3E	5B	40	83
A	B3	69	57	1F	7	1C	8A	BC	20	EB	CE	8E	AB	EE	31	A2
B	73	F9	CA	3A	1A	FB	0D	C1	FE	FA	F2	6F	BD	96	DD	43
C	52	B6	8	F3	AE	BE	19	89	32	26	B0	EA	4B	64	84	82
D	6B	F5	79	BF	1	5F	75	63	1B	23	3D	68	2A	65	E8	91
E	F6	FF	13	58	F1	47	0A	7F	C5	A7	E7	61	5A	6	46	44
F	42	4	A0	DB	39	86	54	AA	8C	34	21	8B	F8	0C	74	67

Підстановка π3

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	68	8D	CA	4D	73	4B	4E	2A	D4	52	26	B3	54	1E	19	1F
1	22	3	46	3D	2D	4A	53	83	13	8A	B7	D5	25	79	F5	BD
2	58	2F	0D	2	ED	51	9E	11	F2	3E	55	5E	D1	16	3C	66
3	70	5D	F3	45	40	CC	E8	94	56	8	CE	1A	3A	D2	E1	DF
4	B5	38	6E	0E	E5	F4	F9	86	E9	4F	D6	85	23	CF	32	99
5	31	14	AE	EE	C8	48	D3	30	A1	92	41	B1	18	C4	2C	71
6	72	44	15	FD	37	BE	5F	AA	9B	88	D8	AB	89	9C	FA	60
7	EA	BC	62	0C	24	A6	A8	EC	67	20	DB	7C	28	DD	AC	5B
8	34	7E	10	F1	7B	8F	63	A0	5	9A	43	77	21	BF	27	9
9	C3	9F	B6	D7	29	C2	EB	C0	A4	8B	8C	1D	FB	FF	C1	B2
A	97	2E	F8	65	F6	75	7	4	49	33	E4	D9	B9	D0	42	C7
B	6C	90	0	8E	6F	50	1	C5	DA	47	3F	CD	69	A2	E2	7A
C	A7	C6	93	0F	0A	6	E6	2B	96	A3	1C	AF	6A	12	84	39
D	E7	B0	82	F7	FE	9D	87	5C	81	35	DE	B4	A5	FC	80	EF
E	CB	BB	6B	76	BA	5A	7D	78	0B	95	E3	AD	74	98	3B	36
F	64	6D	DC	F0	59	A9	4C	17	7F	91	B8	C9	57	1B	E0	61

Підстановка $_{-1}\pi 0$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A4	A2	A9	C5	4E	C9	3	D9	7E	0F	D2	AD	E7	D3	27	5B
1	E3	A1	E8	E6	7C	2A	55	0C	86	39	D7	8D	B8	12	6F	28
2	CD	8A	70	56	72	F9	BF	4F	73	E9	F7	57	16	AC	50	C0
3	9D	B7	47	71	60	C4	74	43	6C	1F	93	77	DC	CE	20	8C
4	99	5F	44	1	F5	1E	87	5E	61	2C	4B	1D	81	15	F4	23
5	D6	EA	E1	67	F1	7F	FE	DA	3C	7	53	6A	84	9C	CB	2
6	83	33	DD	35	E2	59	5A	98	A5	92	64	4	6	10	4D	1C
7	97	8	31	EE	AB	5	AF	79	A0	18	46	6D	FC	89	D4	C7
8	FF	F0	CF	42	91	F8	68	0A	65	8E	B6	FD	C3	EF	78	4C
9	CC	9E	30	2E	BC	0B	54	1A	A6	BB	26	80	48	94	32	7D
A	A7	3F	AE	22	3D	66	AA	F6	0	5D	BD	4A	E0	3B	B4	17
B	8B	9F	76	B0	24	9A	25	63	DB	EB	7A	3E	5C	B3	B1	29
C	F2	CA	58	6E	D8	A8	2F	75	DF	14	FB	13	49	88	B2	EC
D	E4	34	2D	96	C6	3A	ED	95	0E	E5	85	6B	40	21	9B	9
E	19	2B	52	DE	45	A3	FA	51	C2	B5	D1	90	B9	F3	37	C1
F	0D	BA	41	11	38	7B	BE	D0	D5	69	36	C8	62	1B	82	8F

Підстановка $_{-1}\pi 1$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	83	F2	2A	EB	E9	BF	7B	9C	34	96	8D	98	B9	69	8C	29
1	3D	88	68	6	39	11	4C	0E	A0	56	40	92	15	BC	B3	DC
2	6F	F8	26	BA	BE	BD	31	FB	C3	FE	80	61	E1	7A	32	D2
3	70	20	A1	45	EC	D9	1A	5D	B4	D8	9	A5	55	8E	37	76
4	A9	67	10	17	36	65	B1	95	62	59	74	A3	50	2F	4B	C8
5	D0	8F	CD	D4	3C	86	12	1D	23	EF	F4	53	19	35	E6	7F
6	5E	D6	79	51	22	14	F7	1E	4A	42	9B	41	73	2D	C1	5C
7	A6	A2	E0	2E	D3	28	BB	C9	AE	6A	D1	5A	30	90	84	F9
8	B2	58	CF	7E	C5	CB	97	E4	16	6C	FA	B0	6D	1F	52	99
9	0D	4E	3	91	C2	4D	64	77	9F	DD	C4	49	8A	9A	24	38
A	A7	57	85	C7	7C	7D	E7	F6	B7	AC	27	46	DE	DF	3B	D7
B	9E	2B	0B	D5	13	75	F0	72	B6	9D	1B	1	3F	44	E5	87
C	FD	7	F1	AB	94	18	EA	FC	3A	82	5F	5	54	DB	0	8B
D	E3	48	0C	CA	78	89	0A	FF	3E	5B	81	EE	71	E2	DA	2C
E	B8	B5	CC	6E	A8	6B	AD	60	C6	8	4	2	E8	F5	4F	A4
F	F3	C0	CE	43	25	1C	21	33	0F	AF	47	ED	66	63	93	AA

Підстановка π_2^{-1}

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	45	D4	0B	43	F1	72	ED	A4	C2	38	E6	71	FD	B6	3A	95
1	50	44	4B	E2	74	6B	1E	11	5A	C6	B4	D8	A5	8A	70	A3
2	A8	FA	5	D9	97	40	C9	90	98	8F	DC	12	31	2C	47	6A
3	99	AE	C8	7F	F9	4F	5D	96	6F	F4	B3	39	21	DA	9C	85
4	9E	3B	F0	BF	EF	6	EE	E5	5F	20	10	CC	3C	54	4A	52
5	94	0E	C0	28	F6	56	60	A2	E3	0F	EC	9D	24	83	7E	D5
6	7C	EB	18	D7	CD	DD	78	FF	DB	A1	9	D0	76	84	75	BB
7	1D	1A	2F	B0	FE	D6	34	63	35	D2	2A	59	6D	4D	77	E7
8	8E	61	CF	9F	CE	27	F5	80	86	C7	A6	FB	F8	87	AB	62
9	3F	DF	48	0	14	9A	BD	5B	4	92	2	25	65	4C	53	0C
A	F2	29	AF	17	6C	41	30	E9	93	55	F7	AC	68	26	C4	7D
B	CA	7A	3E	A0	37	3	C1	36	69	66	8	16	A7	BC	C5	D3
C	22	B7	13	46	32	E8	57	88	2B	81	B2	4E	64	1C	AA	91
D	58	2E	9B	5C	1B	51	73	42	23	1	6E	F3	0D	BE	3D	0A
E	2D	1F	67	33	19	7B	5E	EA	DE	8B	CB	A9	8C	8D	AD	49
F	82	E4	BA	C3	15	D1	E0	89	FC	B1	B9	B5	7	79	B8	E1

Підстановка π_3^{-1}

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	B2	B6	23	11	A7	88	C5	A6	39	8F	C4	E8	73	22	43	C3
1	82	27	CD	18	51	62	2D	F7	5C	0E	3B	FD	CA	9B	0D	0F
2	79	8C	10	4C	74	1C	0A	8E	7C	94	7	C7	5E	14	A1	21
3	57	50	4E	A9	80	D9	EF	64	41	CF	3C	EE	2E	13	29	BA
4	34	5A	AE	8A	61	33	12	B9	55	A8	15	5	F6	3	6	49
5	B5	25	9	16	0C	2A	38	FC	20	F4	E5	7F	D7	31	2B	66
6	6F	FF	72	86	F0	A3	2F	78	0	BC	CC	E2	B0	F1	42	B4
7	30	5F	60	4	EC	A5	E3	8B	E7	1D	BF	84	7B	E6	81	F8
8	DE	D8	D2	17	CE	4B	47	D6	69	6C	19	99	9A	1	B3	85
9	B1	F9	59	C2	37	E9	C8	A0	ED	4F	89	68	6D	D5	26	91
A	87	58	BD	C9	98	DC	75	C0	76	F5	67	6B	7E	EB	52	CB
B	D1	5B	9F	0B	DB	40	92	1A	FA	AC	E4	E1	71	1F	65	8D
C	97	9E	95	90	5D	B7	C1	AF	54	FB	2	E0	35	BB	3A	4D
D	AD	2C	3D	56	8	1B	4A	93	6A	AB	B8	7A	F2	7D	DA	3F
E	FE	3E	BE	EA	AA	44	C6	D0	36	48	70	96	77	24	53	DF
F	F3	83	28	32	45	1E	A4	D3	A2	46	6E	9C	DD	63	D4	9D

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Юрій Євгенійович Яремчук
Ольга Володимирівна Салієва
Ірина Олексіївна Бондаренко

ОСНОВИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

Навчальний посібник

Рукопис оформила *І. Бондаренко*

Редактор *В. Дружиніна*

Оригінал-макет виготовила *Т. Старічек*

Підписано до видання 17.12.2024.
Гарнітура Times New Roman.
Зам. № Р2024-200.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: irvc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.