МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-06

Дипломний проект

здобувачки освіти денної форми навчання
КГ.06.27.000.ДП

ТІМОФЄЄВОЇ
ВАЛЕРІЇ ЛЕОНІДІВНИ

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітня програма: «**Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-06**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Розробка моделі кодеку на базі алгоритму симетричного шифрування

Проектний матеріал складається з пояснювальної записки на 71 сторінках та графічного (презентаційного) матеріалу на 18 аркушах (слайдах).

Дипломник _____ (Тімофєєва В.Л.)

Керівник _____ (Кривченко Ю.В.)

Консультанти:

з економічної частини _____ (Копайгородська Т.Г.)

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко А.А.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «23» червня 2023 р. Протокол ДКК № 5

Оцінка ДКК 5 (відмінно)

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ІІІ**

Спеціальність **123 «Комп'ютерна інженерія»**

Освітня програма **«Комп'ютерна графіка і Web-дизайн»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР _____

Беркань І.В.

"_____" _____ 2023 р.

ЗАВДАННЯ

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти Тімофєєвої Валерії Леонідівні

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка моделі кодеку на базі алгоритму симетричного шифрування

затверджена наказом по коледжу від "17" жовтня 2022 р. № 235-А2-ОД

2. Термін здачі закінченого проекту (роботи) 12.06.2023

3. Вихідні данні до проекту (роботи) 1. Реалізувати модель криптостійкого кодеку на базі симетричного потокового шифру ChaCha з перевіркою на цілісність даних. 2. Передбачити використання покращеного генератора випадкових чисел для генерування ключу та вектору ініціалізації (вихорю Мерсена, методу Блум-Блум-Шуба); 3. Використовувати візуальний інтерфейс користувача для створюваного додатку; 4. Використовувати алгоритм SHA-512 для хешування; 5. Передбачити перевірку роботи кодеку за допомогою моделювання у Cryptool2

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

Аналіз властивостей алгоритмів симетричного шифрування; Порівняльний огляд сучасних алгоритмів шифрування; Аналіз технічного завдання і постановка задачі проектування; Розробка моделі кодеку та його програмна реалізація; Вибір програмних засобів розробки Вибір програмних складових програмного проекту; Моделювання роботи застосованих алгоритмів; Розробка програмного коду та створення інтерфейсу

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Узагальнена схема симетричного шифрування; Принципи потокового шифрування; Принцип хешування; Генерування псевдовипадкового ключу; Блок-схеми алгоритму, на якому заснований кодек ChaCha; Цикл роботи алгоритму ChaCha; Спрощена схема вихорю Мерсенна; Схема реалізації алгоритму BBS; Схема реалізації алгоритму Меркла-Демгарда; Структура алгоритму SHA-512; Результат моделювання за алгоритмом SHA-512; Результат моделювання шифрування за алгоритмом ChaCha; Головне вікно програми-кодеку на базі алгоритму ChaCha; Приклад дешифрування програмою-кодеком повідомлення

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	Кривченко Ю.В.	1.05	12.06
розділ			
2.Екон. частина	Копайгородська Т.Г.		
3.Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		

7. Дата видачі завдання 1 .05.2023

Керівник Кривченко Ю.В. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	Вступ. Постановка задачі проектування	22.05.2023	виконано
2.	Аналіз технічного завдання та пошук літератури	24.05.2023	виконано
3.	Аналіз властивостей алгоритмів симетричного		
	шифрування	25.05.2023	виконано
4.	Порівняльний огляд сучасних алгоритмів шифрування	29.05.2023	виконано
5.	Визначення найкращих потових алгоритмів шифрування	30.05.2023	виконано
6.	Аналіз технічного завдання і постановка задачі		
	проектування	31.05.2023	виконано
7.	Розробка моделі кодеку та його програмна реалізація	2.06.2023	виконано
8.	Вибір програмних засобів розробки	5.06.2023	виконано
9.	Вибір програмних складових програмного проекту	6.06.2023	виконано
10.	Моделювання роботи застосованих алгоритмів	7.06.2023	виконано
11.	Розробка програмного коду та створення інтерфейсу	8.06.2023	виконано
12.	Економічні розрахунки і розробка питань охорони праці	9.06.2023	виконано
13.	Виконання графічної частини проекту	11.06.2023	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 06. 27 000. 00 ДП	Дипломний проект		
А4			КГ 06. 27 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 06. 27 000. 00 ДП		
Розробив		Тімофєєва В.Л.			Розробка моделі кодеку на базі алгоритму симетричного шифрування	Літ.	Аркуш 3 / Аркушів 71
Перевірив		Кривченко Ю.В.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4КГ-06	
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Інформація грає вирішальну роль як в процесі економічного розвитку, так і в ході конкурентної боротьби на національному і міжнародному ринках. Протиборство розвернулося за перевагу в тих областях, які визначають спрямування науково-технічного прогресу.

Конфіденційна інформація має найбільшу цінність. Люди вже давно навчилися оцінювати важливість інформації і важливість збереження її в таємниці. Зрозуміло, що інформація може бути скарбом, дорожчим за золото, а володіння нею може забезпечити добробут, вплив і владу. Підраховано, що втрата банком 20-25% конфіденційній інформації веде до його банкрутства. Всім відомі численні випадки, коли через втрату інформації люди, зокрема високопоставлені, втрачали свободу і навіть життя.

Рішення для запобігання втрати інформації надає криптографія. На даний час існує багато алгоритмів, які мають захищати дані, але всі вони мають різну надійність. DES (Data Encryption Standard) є симетричним алгоритмом шифрування даних, був стандартом шифрування, прийнятим урядом США, з часом набув міжнародного застосування. DES дав поштовх сучасним уявленням про блочні алгоритми шифрування та криптоаналіз. На сьогоднішній час стандарт DES ще досі використовується певними системами, він має також нові модифікації і ще залишається актуальним для захисту інформації [1].

Криптографічні перетворювачі мають забезпечити математичні методи захисту конфіденційних повідомлень. Навіть у разі їх перехоплення зловмисниками і обробки будь-якими способами з використанням самих швидкодіючих суперкомп'ютерів і останніх досягнень науки і техніки смисловий зміст повідомлень може бути розкритий тільки в перебігу заданого часу, наприклад протягом декількох днів.

У даній дипломній роботі виконується розробка моделі кодеку симетричного потокового алгоритму шифрування ChaCha з застосуванням рівномірного розподілу. Упровадження цієї системи дозволить збільшити швидкість обробки даних для шифрування із забезпеченням захисту від зловмисників.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Аналіз властивостей алгоритмів симетричного шифрування

Існує велика кількість методів шифрування. Головним чином ці методи діляться на симетричні та асиметричні, в залежності від структури використовуваних ключів. Крім того, методи шифрування можуть мати різну криптостійкість і по-різному обробляти вхідні дані – блокові шифри і потокові шифри. Ключ є фундаментальною частиною захисту конфіденційності інформації, повідомлення або частини даних. Процес шифрування та дешифрування може бути ініційований лише за допомогою ключа. Ключ є невід'ємною частиною процесу кодування і декодування даних. Ключі шифрування розроблені таким чином, щоб вони були абсолютно унікальні, використовуючи набір різних алгоритмів. Ключ шифрування використовується для кодування або декодування даних. Як правило, ключ є випадковим двійковим або фактичним паролем. У зв'язку з тим, що алгоритми є загальнодоступними і доступні для будь-кого, якщо хакер отримає ключ шифрування, зашифровані дані легко розшифруються до відкритого тексту.

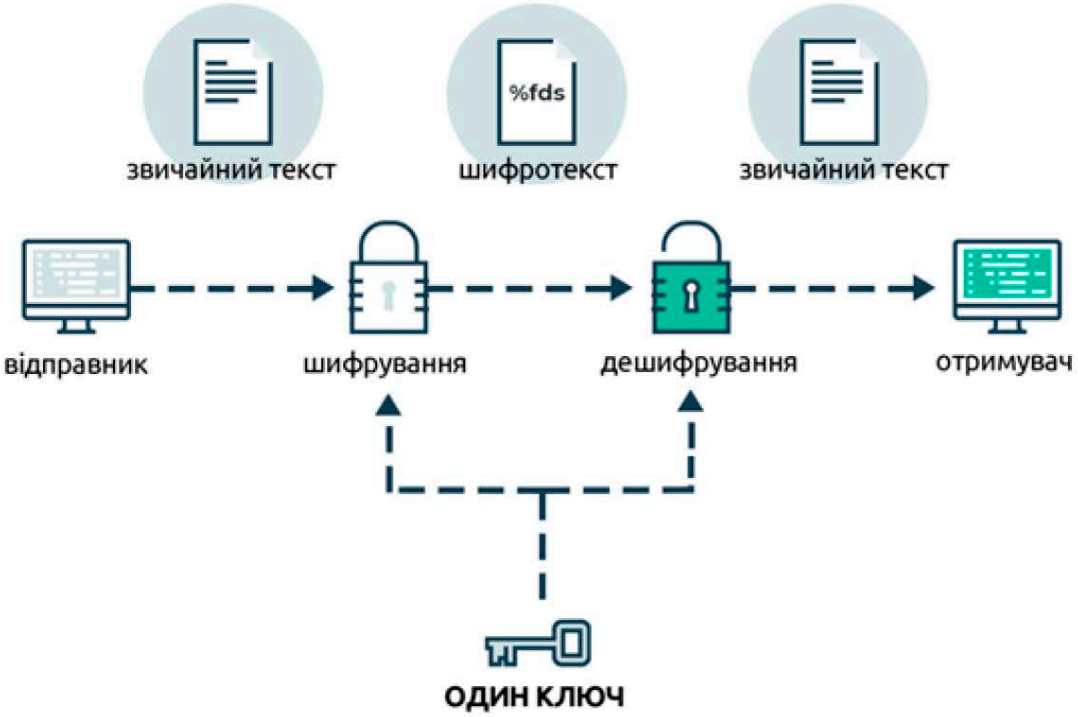


Рисунок 1.1. Узагальнена схема симетричного шифрування

Системи симетричного шифрування використовують один і той самий ключ для кодування і декодування інформації (рис. 1.1). Його найкраще використовувати

при обміні невеликих об'ємів даних один з одним. Симетричне шифрування набагато швидше ніж асиметричне, але відправник має обмінятися ключем шифрування з одержувачем, перш ніж останній зможе розшифрувати повідомлення.

Потокові алгоритми шифрування шифрують та дешифрують кожний окремий біт (рис. 1.2).

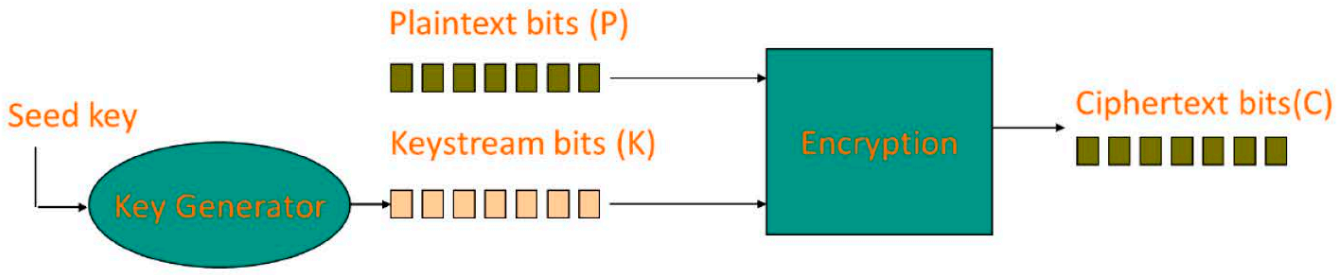


Рисунок 1.2. Принципи потокового шифрування

У порівнянні з блочним даний метод є більш складним, але швидшим, оскільки не витрачає час на буферизацію даних з вводу. Основою цього методу є зміщення як у шифрі Цезаря, Плейфера, Гілла, підстановочному та моноалфавітному шифрах. Використовують ключ лише один раз. Такі шифри знайшли своє застосування в апаратному забезпеченні, безпечному з'єднанні SSL для Інтернету. Приклади сучасних алгоритмів, що застосовують потокове шифрування є OFB та CFB [1].

Блочні шифри приймають певну кількість байтів або блоків та працюють з ними, перетворюючи в єдину одиницю нової інформації (рис. 1.3).

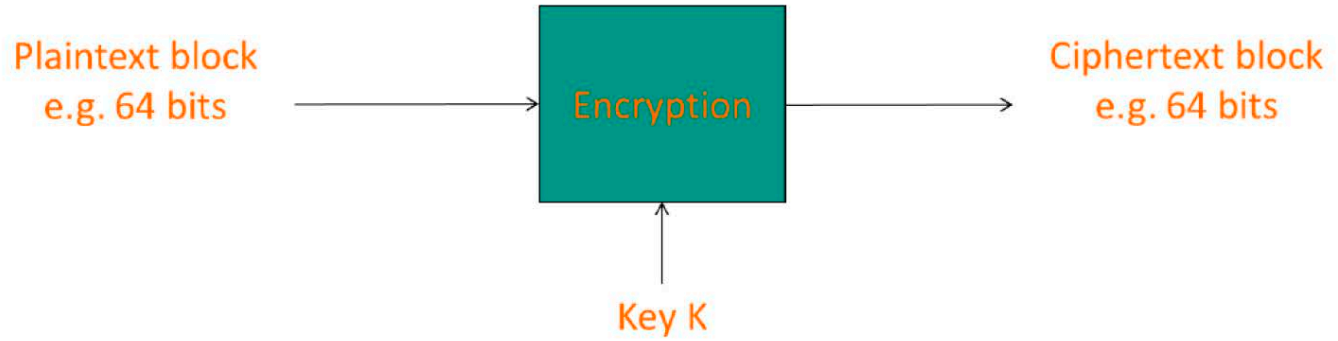


Рисунок 1.3. Принципи блочного шифрування

Блочне шифрування бере за основу такі методи транспортування, як шифр Вернама, перестановочний та книжковий шифр. Даний метод можна створювати з потокового, але навпаки операція не можлива. На відміну від потокових, наявна

техніка плутанини, а також дифузія. Такі техніки допомагають впевнитися, що з шифротексту не можна отримати жодного натяку на початковий текст. Шифрування використовує 64 або більше бітів, є простим, але повільним. Воно застосовується у програмному забезпеченні, при кодуванні баз даних та файлів. Приклади сучасних алгоритмів, що застосовують потокове шифрування є CBC та ECB.

Симетричні алгоритми шифрування застосовуються для шифрування початкового тексту, наприклад даних на жорстких дисках. Довгий час найбільш поширеним був алгоритм DES, на зміну якого прийшов ефективний алгоритм AES. Головною перевагою симетричних алгоритмів є те, що вони мають добрі показники швидкості для читання та запису інформації. Проте симетричні алгоритми мають і певні недоліки:

– ключі шифрування не є набором тексту, зрозумілим користувачам, а набір згенерованих даних, що мають надсилатися, абсолютно секретні;

– при отриманні ключа, весь текст, що передається, може бути розшифрованим. У випадку використання асиметричного методу користувач, що отримав приватний ключ, може розшифрувати повідомлення, яке він надсилає, але не може розшифрувати повідомлення, надіслане йому [2].

Асиметричні алгоритми для засекречення інформації використовують загальнодоступний, відкритий ключ, який може отримати будь-хто. Такі ключі застосовуються для шифрування повідомлень особою, яка хоче надіслати інформацію іншій, яка має доступ до цього ключа. Для дешифрування використовується приватний ключ, що впроваджується для встановлення захищеного з'єднання. Регенерація ключа відбувається лише за умови, що буде порушена процедура передачі приватного ключа. Кодування з відкритим асиметричним ключем має свої переваги:

– дозволяє іншій особі переконатися у тому, що надіслані дані саме від конкретного користувача;

– має здатність перевірити чи була зміна відкритого тексту під час передачі.

Проте асиметричні алгоритми мають і певні недоліки:

– не підходить для кодування великих об'ємів інформації через повільну

роботу;

– необхідно перевіряти, чи дійсно вказана особа є власником відкритого ключа;

– при втраті приватного ключа ніхто не зможе дешифрувати повідомлення.

Треба зазначити, що швидкість виконання симетричних алгоритмів вища, ніж асиметричних, але на практиці ці два методи інтегруються разом щоб покращити захист систем від небажаного витоку інформації та забезпечити зберігання ресурсів.

1.1.1 Особливості блокових криптоалгоритмів

Блокові криптоалгоритми перетворюють блок вхідної інформації фіксованої довжини і одержують вихідний блок того ж розміру, але недоступний для читання стороннім особам, що не володіють інформацією про ключ. Таким чином, схему роботи блокового шифру можна описати функціями Z=EnCrypt(X,Key) і X=DeCrypt(Z,Key), де ключ Key є параметром блокового шифру і являє собою деякий блок двійкової інформації фіксованого розміру. Вихідний X і зашифрований Z блоки даних також мають фіксовану розрядність, яка є рівною між собою, але необов'язково дорівнює довжині ключа. Блокові шифри є основою, яку беруть для реалізації практично всі криптосистеми (табл. 1.1). Методика створення ланцюгів із байт, що зашифровані блоковими алгоритмами, дозволяє шифрувати пакети інформації необмеженої довжини. Така властивість блокових шифрів, як швидкість роботи, використовується асиметричними криптоалгоритмами. Відсутність статистичної кореляції між бітами вихідного потоку блокового шифру використовується для обчислення контрольних сум пакетів даних і в хешуванні паролів [2].

Таблиця 1.1. Сучасні симетричні криптоалгоритми

Назва алгоритму	Розмір блоку	Довжина ключа
IDEA	64 біта	128 біт
CAST128	64 біта	128 біт
BlowFish	64 біта	128 – 448 біт
ГОСТ	64 біта	256 біт
TwoFish	128 біт	128 – 256 біт
MARS	128 біт	128 – 1048 біт

До стійких криптоалгоритмів застосовується дуже важлива вимога, яку вони повинні обов'язково задовольняти: при відомих вхідному і вихідному значеннях блоку ключ, за допомогою якого здійснюється шифрування, може бути знайденим тільки шляхом повного перебору. Часто зустрічаються ситуації, в яких сторонньому спостерігачеві відома частина вихідного тексту. Це можуть бути стандартні написи в електронних бланках, фіксовані заголовки форматів файлів, що досить часто зустрічаються в тексті. Таким чином, до функції стійкого блокового шифру Z=EnCrypt(X,Key) висуваються наступні вимоги:

- функція EnCrypt повинна бути оборотною;
- не повинно існувати інших методів прочитання повідомлення X за відомим блоком Z, крім повного перебору ключів Key;
- не повинно існувати інших методів визначення яким ключем Key було зроблене перетворення відомого повідомлення X у Z, окрім повного перебору ключів.

Таблиця 1.2. Бінарні операції шифрування

Бінарні операції	
Додавання	X'=X+V
Виключне АБО	X'=X XOR V
Множення за модулем 2^N+1	X'=(X*V) mod (2^N+1)
Множення за модулем 2^N	X'=(X*V) mod (2^N)
Бітові зсуви	
Арифметичний зсув вліво	X'=X SHL V
Арифметичний зсув вправо	X'=X SHR V
Циклічний зсув вліво	X'=X ROL V
Циклічний зсув вправо	X'=X ROR V
Табличні підстановки	
S-box (англ. substitute)	X'=Table[X,V]

За допомогою деяких методів розробники блокових криптоалгоритмів досягають одночасного виконання цих трьох умов з дуже великою вірогідністю. Всі дії, які здійсненні над даним блоковим криптоалгоритмом, полягають в тому, що перетворений блок може бути представлений у вигляді цілого невід'ємного числа з діапазону, що відповідає його розрядності. Так, наприклад, 32-бітний блок даних можна інтерпретувати як число з діапазону 0..4'294'967'295. Крім того, блок,

розрядність якого є ступенем двійки, можна трактувати як кілька незалежних невід'ємних чисел з меншого діапазону (розглянутий вище 32-бітний блок можна також представити у вигляді 2 незалежних чисел з діапазону 0..65535 або у вигляді 4 незалежних чисел з діапазону 0..255. Блокові криптоалгоритми дозволять здійснювати над цими числами за визначеною схемою дії, зазначені у табл. 1.2.

Для кожного з вказаних вище перетворень параметр V може бути використаний:

- як фіксоване число (наприклад, X'=X+125);
- як число, що отримане за допомогою додавання ключа (наприклад, X'=X+F(Key));
- як число, що отримане із незалежної частини блоку (наприклад, $X2'=X2+F(X_1)$).

Третій варіант використовується в схемі, яка мережею Фейстеля (Feistel). Послідовність операцій, що виконуються над блоком, комбінації перерахованих вище варіантів V і самі функції F і складають "ноу-хау" кожного конкретного блокового криптоалгоритму. Один-два рази в рік дослідницькі центри світу публікують черговий блоковий шифр, що під проведенням атак криптоаналітиків або здобуває статус стійкого криптоалгоритму, або навпаки. Характерною ознакою блокових алгоритмів є багаторазове і непряме використання ключа. Це диктується в першу чергу вимогою неможливості зворотного дешифрування у відношенні ключа при відомих вихідному і шифрованому текстах. Для рішення цієї задачі в приведених вище перетвореннях найчастіше використовується не саме значення ключа або його частини, а деяка необоротна функція від значення ключа. Більш того, в подібних перетвореннях той самий блок або елемент ключа використовується багаторазово. Це дозволяє при виконанні умови оборотності функції щодо величини X зробити функцію необоротною щодо ключа Key [3].

Операція шифрування або дешифрування окремого блоку в процесі кодування пакета інформації виконується багаторазово (іноді до сотень тисяч разів), а значення ключа і, отже, функцій Vi(Key) залишається незмінним, тому іноді стає доцільно заздалегідь однократно обчислити дані значення і зберігати їх в оперативній пам'яті

разом із ключем. Варто зазначити, що дана операція ніяким чином не змінює ні довжину ключа, ні криптостійкість алгоритму в цілому. Тут відбувається лише оптимізація швидкості обчислень шляхом кешування (caching) проміжних результатів. Описані дії зустрічаються практично в багатьох блокових криптоалгоритмах і називаються розширенням ключа (key scheduling).

Подальшою модифікацією описаного вище методу змішування поточної частини блоку, що шифрується, з результатом деякої функції, яка отримується шляхом обчислення від іншої незалежної частини того ж блоку, є мережа Фейстеля. Цей метод одержав широке поширення, оскільки забезпечує виконання вимоги про багаторазове використання ключа і вихідного блоку інформації. Класична мережа Фейстеля має наступну структуру (рис. 1.4).

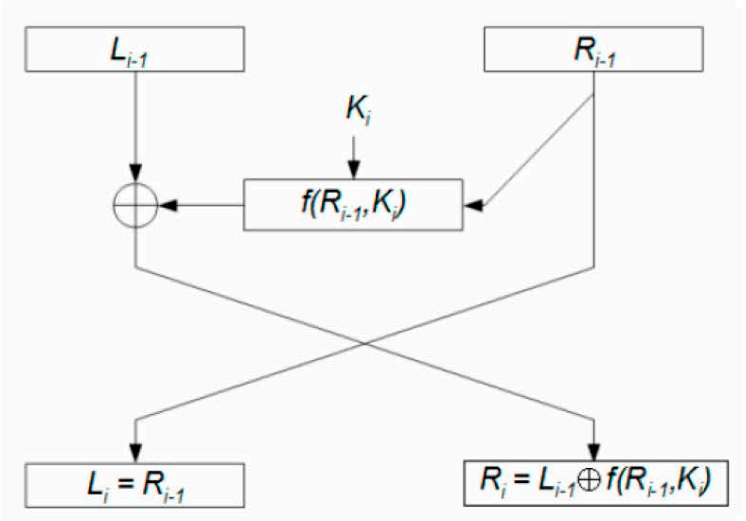


Рисунок 1.4. Структура мережі Фейстеля при шифруванні

Алгоритм шифрування у мережі Фейстеля є таким:

1. Кожен блок даних розбивається на дві рівні частини ліву L) і праву R;

2. Права частина видозмінюється деякою функцією f R K) залежно від раундового ключа K;

3. Здійснюється додавання за модулем 2 лівої частини L та f R K;

4. Результат додавання присвоюється новому правому підблоку, а правий підблок присвоюється без змін новому лівому підблоку (вхідні дані для наступного раунду).

Породжені з вхідного блоку незалежні потоки інформації називаються галузями мережі. У класичній схемі їх дві. Величини Vi називаються параметрами

мережі, звичайно це функції від значення ключа. Функція F називається утворюючою. Дія, що складається з однократного обчислення утворюючої функції і наступного накладення її результату на іншу галузь з обміном їх місцями, називається циклом або раундом (round) мережі Фейстеля. Оптимальне число раундів K – від 8 до 32. Важливо те, що збільшення кількості раундів значно збільшує криптостійкість будь-якого блокового шифру до криптоаналізу. Можливо, ця особливість і вплинула на активне поширення мережі Фейстеля – адже при виявленні, скажімо, якого-небудь слабкого місця в алгоритмі, здебільшого достатньо збільшити кількість раундів на 4-8, не переписуючи сам алгоритм. Часто кількість раундів не фіксується розробниками алгоритму, а лише вказуються розумні межі (обов'язково нижній, і не завжди – верхній) цього параметра. Варто зазначити, що схема є оборотною. Мережа Фейстеля володіє тією властивістю, що навіть якщо утворюючу функцію F буде використано як необоротне перетворення, то й у цьому випадку весь ланцюжок буде відновлено. Це відбувається внаслідок того, що для зворотного перетворення мережі Фейстеля не потрібно обчислювати функцію F-1.

Мережа Фейстеля є симетричною. Використання операції XOR, оборотною своїм же повтором, і інверсія останнього обміну гілок роблять можливим декодування блоку тією ж мережею Фейстеля, але з інверсним порядком параметрів Vi. Зазначимо, що для оборотності мережі Фейстеля не має значення чи є число раундів парним або непарним числом. У більшості реалізацій схеми, в яких обидві перераховані вище умови (операція XOR і знищення останнього обміну) збережені, пряме і зворотне перетворення виробляються однією і тією ж процедурою. З незначними доробками мережу Фейстеля можна зробити й абсолютно симетричною, тобто виконуючі функції шифрування і дешифрування тим самим набором операцій. Математичною мовою це записується як "Функція EnCrypt тотожно дорівнює функції DeCrypt".

Набагато частіше застосовують модифікацію мережі Фейстеля для більшого числа гілок. Це в першу чергу пов'язане з тим, що при великих розмірах блоків (128 і більше біт), що шифруються, стає незручно працювати з математичними

функціями за модулем 64 і вище. Як відомо, основні одиниці інформації, що обробляються процесорами на сьогоднішній день – це байт і подвійне машинне слово 32 біта. Тому все частіше й частіше в блокових криптоалгоритмах зустрічається мережа Фейстеля з 4-ма гілками. Для більш швидкого перемішування інформації між гілками (а це основна проблема мережі Фейстеля з великою кількістю гілок) застосовуються дві модифіковані схеми, що називаються "type-2" і "type-3", зображені на рис. 1.5б і в, відповідно.

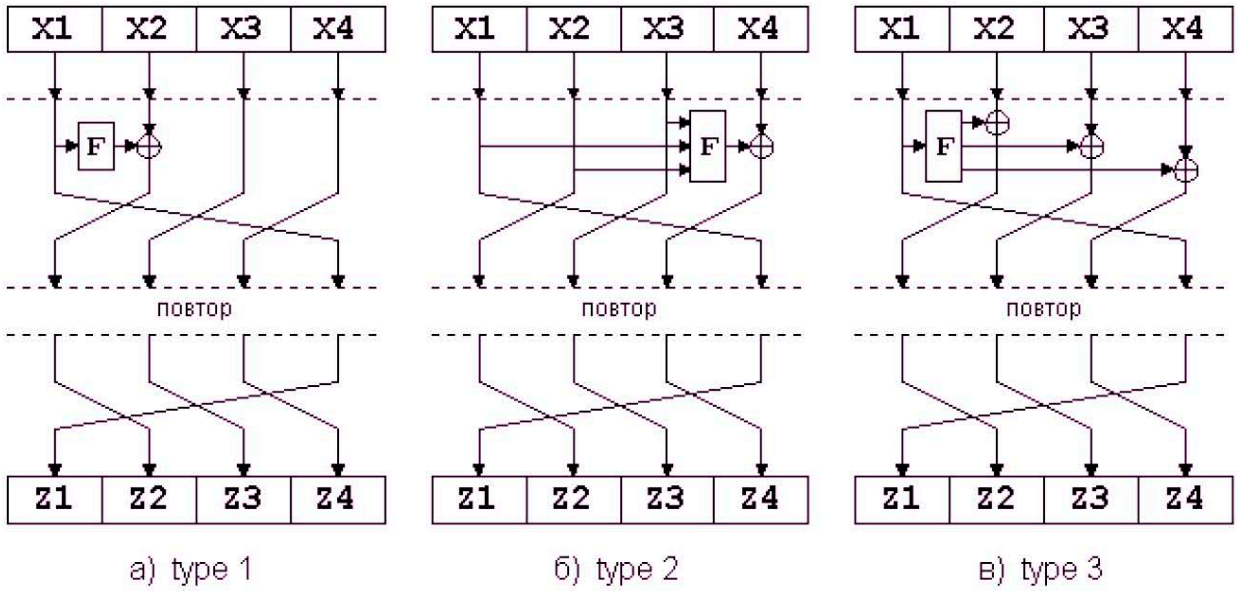


Рисунок 1.5. Принципи модифікації мережі Фейстеля

Мережа Фейстеля надійно зарекомендувала себе як криптостійка схема добутку перетворень і її можна знайти практично в будь-якому сучасному блоковому шифрі. Незначні модифікації стосуються звичайно додаткових початкових і останніх перетворень (whitening) над блоком, що шифрується. Подібні перетворення, що виконуються, звичайно, також за допомогою "виключного АБО" або додавання, мають за мету підвищити початкову рандомізацію вхідного тексту. Таким чином, криптостійкість блокового шифру, що використовує мережу Фейстеля, визначається на 95% функцією F і правилом обчислення Vi із ключа [4].

1.1.2 Аналіз криптографічної стійкості систем шифрування

Стійкість криптографічних систем до злому визначається часом та ресурсами, які буде затрачено на те, аби відновити вихідний або відкритий текст. На сьогоднішній день, якщо використовується ключ довжиною 256 біт, вважається, що

системою захищена добре, проте потужність комп'ютерів зростає, з'являється необхідність покращувати алгоритми й надалі. Основними принципами до систем шифрування є вимоги до його ключа:

- ключ має бути випадковим;
- ключ має передаватися конфіденційно (інформація має бути обмежена);
- ключ має бути за розмірами більшим або таким самим, як повідомлення;
- ключ використовується лише один раз (повторне використання заборонене, після відповідних маніпуляцій ключ знищується);
- вихідне повідомлення не повинно містити жодної інформації щодо вмісту ключа.

Зазначені вимоги актуальні навіть при умові, що у зловмисників є необмежені ресурси в обчислювальній системі і криптоаналіз проводиться у відповідності до теорії ймовірності. Криптографічні системи можуть бути вразливими до криптографічного аналізу у залежності від їх реалізацій, протоколів, алгоритмів які використовуються. Найбільш незламним є шифрування схемою одноразових блокнотів (шифр Вернама) при правильному використані. На сьогодні це єдина система кодування з теоретично доведеною абсолютною криптографічною стійкістю тому, що ця техніка відповідає основним вимогам, які описав Шеннон. Бінарна версія шифру Вернама описана нижче (1.1).

$$k = m = c = \{0,1\}^2 \qquad (1.1)$$

де m – початковий (відкритий) текст; c – шифротекст; k – ключ.

Такий шифр неможливо зламати, оскільки немає критерію, згідно якого можна було б дізнатися, розшифрована дана послідовність, чи ні. Але основним недоліком є те, що ключі та блокноти мають передаватися через системи, які можна зламати та отримати інформацію, яка здатна дешифрувати дані. Проте, головна проблема полягає у генеруванні випадкових чисел та передачі ключа такої ж довжини, як вихідний текст, що використовує багато ресурсів. Відповідний метод схеми одноразових блокнотів заснований на модульному додаванні. Застосування одного й того самого ключа для двох різних повідомлень призводить до вразливостей системи через те, що з'являється виключна диз'юнкція двох текстів (1.2).

$$(m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2 \qquad (1.2)$$

де m_1, m_2 – відкритий текст; k – сформований ключ.

Принцип Керкгоффза, викладений у роботі «Військова криптографія», став ще одним правилом розробки криптостійких систем. Ця робота описує шість основних підходів для проектування надійних шифрів:

– відкритість (кожен має право на доступ до алгоритму);

– є можливість зміни ключа за потребами та його передача має здійснюватися без жодних проблем;

– для обслуговування систем необхідна лише одна людина;

– простота використання;

– система повинна бути криптостійкою (математично та практично не здатна розшифруватися);

– придатність передачі при листуванні.

Архітектура алгоритму шифрування, таким чином, не повинна впливати на криптографічну стійкість. Це досягається відкритим кодом алгоритмів, що дає можливість криптоаналітикам дослідити вразливості методу та виправити відповідні помилки, адже з плином часу будь-які системи можуть бути зламані хакерами [4].

1.1.3 Забезпечення цілісності даних

Криптографічна система має забезпечувати не тільки конфіденційність, а й цілісність даних, тому що навіть криптостійкі алгоритми не можуть захистити від пошкодження інформації сторонніми особами. Автентичність даних (цілісність) означає, що отримувач при наявності ключа може перевірити справжність даних. Базові методи автентифікованого шифрування є такими:

– SSL – використовує математичне шифрування, порт 443, тобто створює безпечний канал передачі між двома системами (зазвичай клієнт та сервер), що дає змогу застосовувати цей метод для відправок особистих даних з банків, інформації про кредитні картки, доступу до віддалених програм;

– SSH – використовує порт 22, що дає змогу підключитися до іншого

комп'ютера. Призначений для виконання команд через віддалений доступ, в той час як SSL призначений для передачі інформації. Може використовуватися лише для TCP;

– IPSec – забезпечує захист інформації, цілісність, автентифікацію джерела та працює на мережевому рівні TCP/IP. Надає віддаленому комп'ютеру доступ до всієї центральної мережі. Може використовуватися як для UDP, так і для TCP.

Той чи інший протокол використовують в залежності від ситуацій. Якщо кількість програм у системі велика, застосовують IPSec із захистом пакетів та приховуванням пункту призначення, а якщо мала, то застосовують автентифікацію за допомогою SSL або SSH [5].

1.1.4 Застосування хеш-функції

Хеш-функція є математичною функцією, яка на вході приймає інформацію та перетворює її у стиснену версію фіксованої довжини інших числових даних. Так, файли розміром у декілька гігабайтів та у декілька байтів будуть перетворюватися в однакову за розміром послідовність.

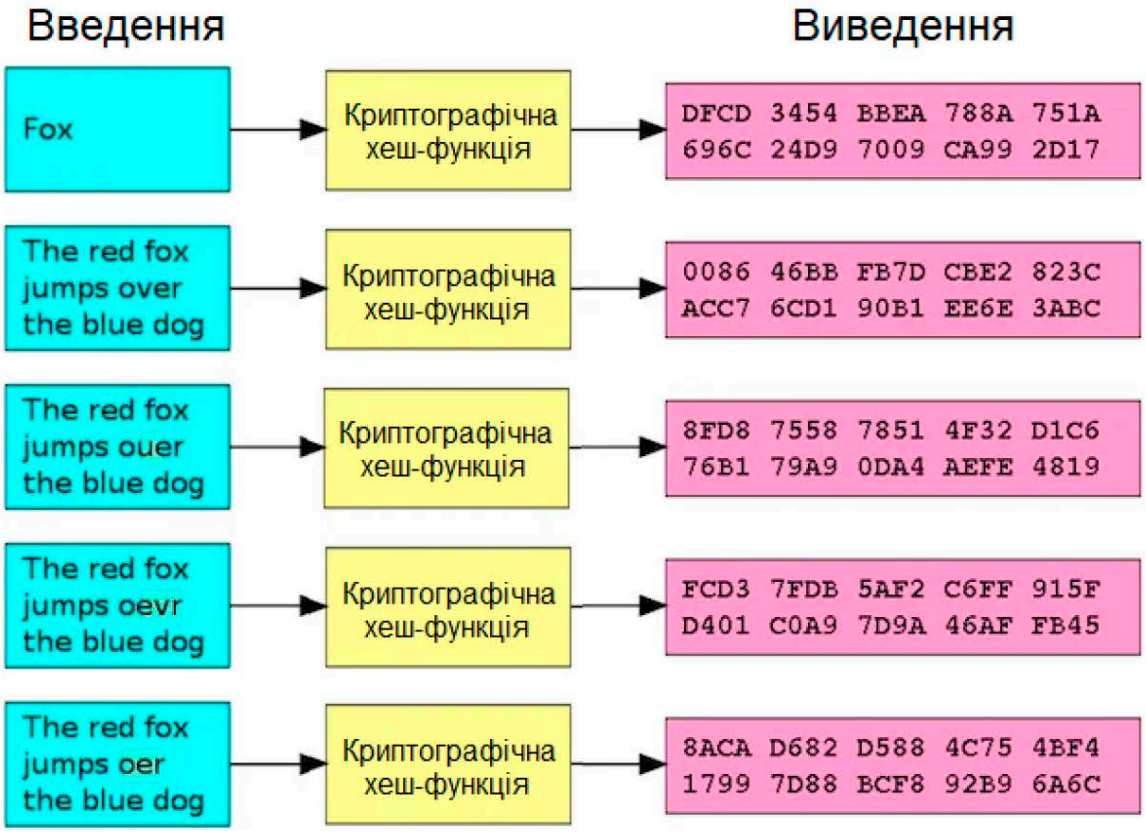


Рисунок 1.6. Принцип хешування за допомогою хеш-функції

Хеш-функції обчислюються швидше, ніж виконується симетричне шифрування. Проте різниця між ними у тому, що хеш-функція є одностороннім криптографічним алгоритмом, який є не оборотним. Це пояснюється тим, що невідома початкова довжина вхідного тексту. Перевагою є те, що зловмисник, отримавши пароль у вигляді хешу, не може увійти у систему, але недолік – дані назад перетворитися не здатні. Найбільше застосування хеш-функції отримали у перевірках цілісності інформації, алгоритм генерує контрольну суму у файлах, у електронних підписах та при зберіганні паролів, але немає гарантій щодо оригінальності файлу, адже зловмисники, замість того щоб змінити частину тексту, можуть замінити його цілком.

1.1.5 Застосування генераторів випадкових чисел

Змінна, значення якої не можна передбачити, є випадковою величиною. Послідовність таких чисел не можна описати жодною формулою, тобто вона абсолютну інформаційну ентропію (1.3).

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \qquad (1.3)$$

де X – випадкова величина; x_i – можливі значення; p_i – ймовірність події.

Випадкові величини діляться на дискретні і неперервні. Дискретні величини мають такі особливості:

- ймовірності знаходяться між 0 та 1, сума яких дорівнює 1;
- кількість можливих значень можна пронумерувати, тобто кількість або скінченна, або нескінченна зліченна;
- дані можна отримати шляхом підрахунку.

Неперервні величини мають такі особливості:

- приймають всі значення на заданому скінченному чи нескінченному інтервалі;
- розподіл ймовірностей описується кривою щільності;
- кількість можливих значень є нескінченною;
- дані можна отримати шляхом вимірювання.

Одним з основних понять у теорії ймовірностей є математичне сподівання, яке

обчислюється множенням кожного з можливих результатів на їх ймовірності. Для дискретної випадкової величини математичне сподівання описується за формулою (1.4).

$$E(X) = \sum_{i=1}^{\infty} p_i x_i \quad (1.4)$$

де X – випадкова величина; x_i – можливі значення; p_i – ймовірності події.

Для неперервної випадкової величини математичне сподівання описується за формулою (1.5).

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx \quad (1.5)$$

де X – випадкова величина; $f(x)$ – густина розподілу.

Якщо генератор випадкових чисел на послідовності (0;1) видає числа з рівномірним розподілом, він є ідеальним. За кожну спробу отримати значення на виході буде отримано лише одне випадкове число, тобто ймовірності зустріти будь-яке число є однаковими. Оскільки комп'ютери використовують алгоритмічні методи формування послідовностей випадкових значень, вони залежні від прописаних установок, що робить таку генерацію псевдовипадковою або квазі-випадковою – вони мають низьку ентропію. Найбільшим плюсом цього методу є використання мінімуму ресурсів.

Існують тести, що можуть визначити, наскільки задана послідовність випадкова. Колмогоровська складність – це міра обчислювальної здатності, яка необхідна для точного визначення певного об'єкту, наприклад тексту.

$$K_f(s) = \min\{|p| : f(p) = s\} \quad (1.6)$$

де s – рядок з даними; f – машина Тюрінга.

Проблемою оцінки за Колмогоровською складністю є те, що таку функцію K_f неможливо обчислити через те, що поки не існує ефективних способів для її представлення. По суті, ця складність – довжина кінцевої стиснутої версії даних, які необхідні для відновлення деякої інформації. У вивченні випадковостей також застосовується метод Монте-Карло на основі закону великих чисел, що допомагає вирішувати ймовірнісні задачі, застосовуючи статистику. Цей метод дозволяє визначити доцільність взяття величини X через систему проведення великої

кількості випробувань. Основною проблемою є генерація незалежних випадкових чисел та повільна збіжність у результатах. Для криптостійких систем потрібно наближати здатність комп'ютерів не бути детермінованими і застосовувати абсолютну ентропію у ключах та алгоритмах. На разі використовуються криптографічно стійкий генератор псевдовипадкових чисел (Cryptographically secure pseudorandom number generator, CSPRNG), які приймають на вхід початкове випадкове значення (seed), а згодом підвищують ймовірність формування послідовностей, які є випадковими [5].

1.1.6 Запобігання частотному аналізу

При криптоаналізі одним з основних методів є застосування частотного аналізу. Для унеможливлення даного варіанту злому алгоритмів передбачено дискретний рівномірний розподіл (рис. 1.7), коли кінцева кількість результатів виконання процесу має однакову ймовірність.

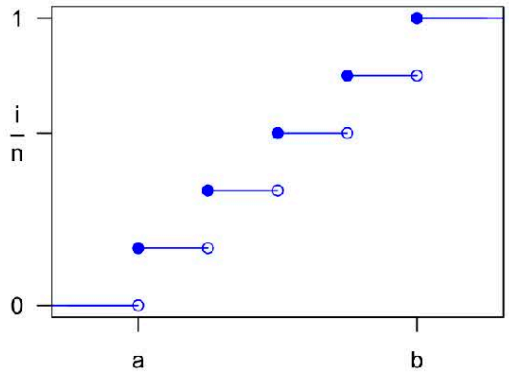


Рисунок 1.7. Кумулятивна функція дискретного рівномірного розподілу для n = 5

Прикладом дискретного рівномірного розподілу є кидання грального кубику, де ймовірність випадіння кожної сторони становить 1/6. Рівномірний дискретний розподіл описується так:

$$f(x_i) = \frac{1}{n} \qquad (1.7)$$

де n – кількість випробувань; x – значення.

Дискретного рівномірного розподілу важко досягнути у детермінованих обчислювальних системах, тому можна застосувати метод Alias, який працює на двох таблицях: ймовірностей та псевдонімів. Використовується цей метод для даних, що мають незмінну таблицю псевдонімів.

1.2 Порівняльний огляд сучасних алгоритмів шифрування

Сучасні алгоритми шифрування у переважній більшості застосовують операцію XOR, тобто складання за модулем 2.

1.2.1 Алгоритм XOR

Шифр XOR (рис.1.8) є алгоритмом шифрування, який у якості ключу використовує ключове слово та може бути записаний формулою

$$Ci = Pi\ XOR\ Kj \qquad (1.8)$$

де Kj – j-та літера ключового слова, представлена в кодуванні ASCII. Ключове слово повторюється поки не отримано гаму, рівну довжині повідомлення.

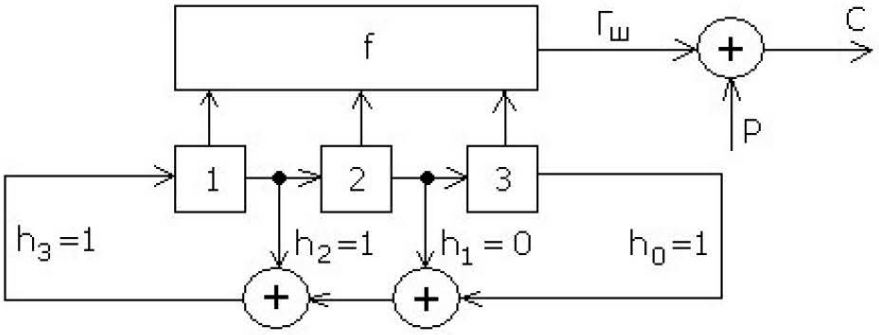


Рисунок 1.8. Реалізація шифру XOR

В якості ключового потоку можна використовувати нелінійно відфільтрований вміст зсувного регістру, а для отримання послідовності максимальної довжини – лінійний зворотний зв'язок.

1.2.2 Алгоритм DES і його модифікації

Алгоритм DES (Data Encryption Standard) розроблений фірмою IBM і був затверджений урядом США як офіційний стандарт шифрування. Ключем в цьому алгоритмі шифрування є 56 бітний блок + 8 біт контролю парності, що розміщені в позиціях 8, 16, 24, 32, 40, 48, 56, 64 (використовується при знаходженні помилок при обміні та зберіганні ключів). Вихідний текст є блоком 64 біт. Процес шифрування складається з:

- початкової перестановки;
- 16-ти раундів шифрування;
- кінцевої перестановки.

Зашифрований текст також є блоком 64 біт. Загальна схема алгоритму DES наведена на рис.1.9.

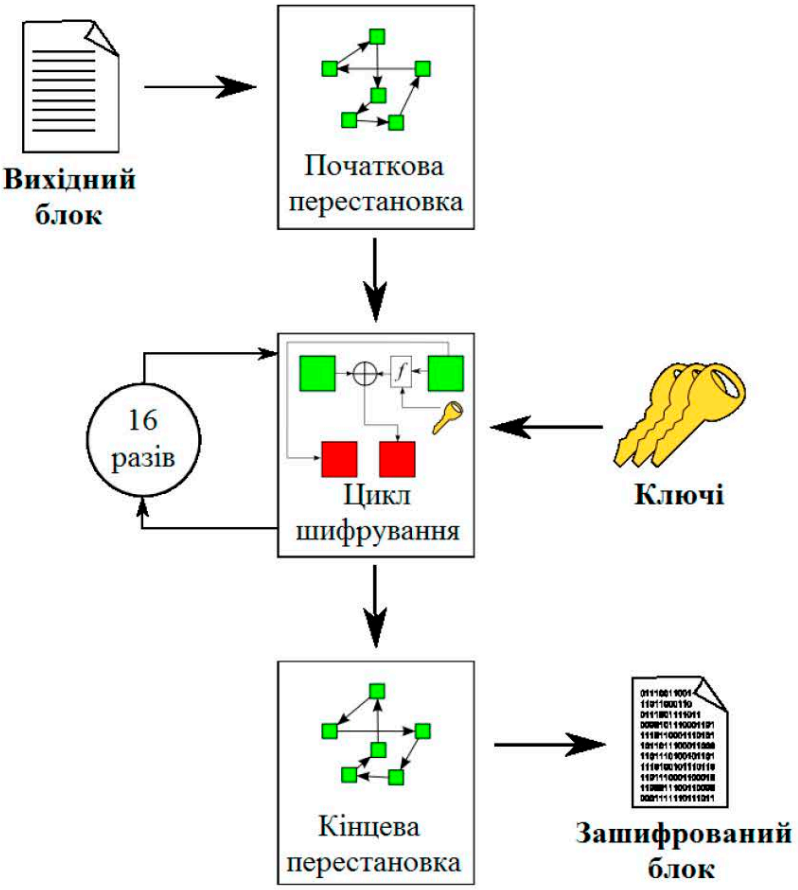


Рисунок 1.9. Загальна схема алгоритму DES

Початковий текст T (блок 64 біт) перетворюється за допомогою початкової перестановки IP (Initial Permutation) за таблицею початкової перестановки (табл.1.3).

Таблиця 1.3. Початкова перестановка 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

Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти з у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі

виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці [6].

Основним недоліком DES вважається маленька довжина ключа, тому вже давно почали розроблятися різні альтернативи цьому алгоритму шифрування. Один з підходів полягає в тому, щоб розробити новий алгоритм, і успішний тому приклад – IDEA. Інший підхід припускає повторне застосування шифрування за допомогою DES з використанням декількох ключів. Найпростіший спосіб збільшити довжину ключа складається в повторному застосуванні DES із двома різними ключами. Використовуючи незашифроване повідомлення P і два ключі K_1 і K_2, зашифроване повідомлення C можна одержати в такий спосіб:

$$C = E_{k2} [E_{k1} [P]] \qquad (1.9)$$

Для дешифрування потрібно, щоб два ключі застосовувалися у зворотному порядку:

$$P = D_{k1} [D_{k2} [C]] \qquad (1.10)$$

У цьому випадку довжина ключа дорівнює 56 * 2 = 112 біт.

Очевидна протидія атаці "зустріч посередині", що буде розглядатися пізніше, складається у використанні третьої стадії шифрування із трьома різними ключами. Це піднімає вартість лобової атаки до 2^{168}, що на сьогоднішній день вважається вище практичних можливостей. Але при цьому довжина ключа дорівнює 56 * 3 = 168 біт, що іноді буває громіздко. Як альтернатива є метод потрійного шифрування, що використовує тільки два ключі. У цьому випадку виконується послідовність зашифрування-розшифрування-зашифрування (EDE).

$$C = E_{K1} [D_{K2} [E_{K1} [P]]] \qquad (1.11)$$

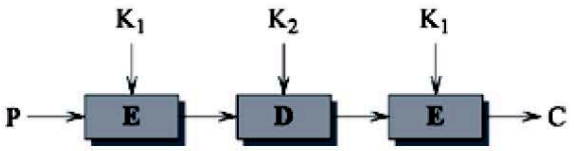


Рисунок 1.10. Шифрування потрійним DES

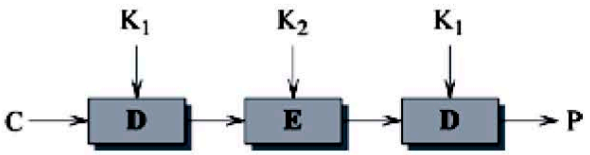


Рисунок 1.11. Дешифрування потрійним DES

Не має великого значення, що використовується на другій стадії: шифрування або дешифрування. У випадку використання дешифрування існує тільки та перевага, що можна потрійний DES звести до звичайного одиночного DES, використовуючи $K_1 = K_2$:

$$C = E_{K1} [D_{K1} [E_{K1} [P]]] = E_{K1} [P] \qquad (1.12)$$

Потрійний DES є досить популярною альтернативою DES і використовується при керуванні ключами в стандартах ANSI X9.17 і ISO 8732 і в PEM (Privacy Enhanced Mail). Відомих криптографічних атак на потрійний DES не існує. Ціна підбора ключа в потрійному DES дорівнює 2^{112}.

1.2.3 Алгоритм AES

Шифр AES є алгоритмом блочного шифрування з відкритим ключем, що використовує 128-бітні дані та ключі різної довжини. Масив байтів записуються у таблицю, де виконуються стандартні математичні операції над окремими елементами масиву (рис. 1.12).

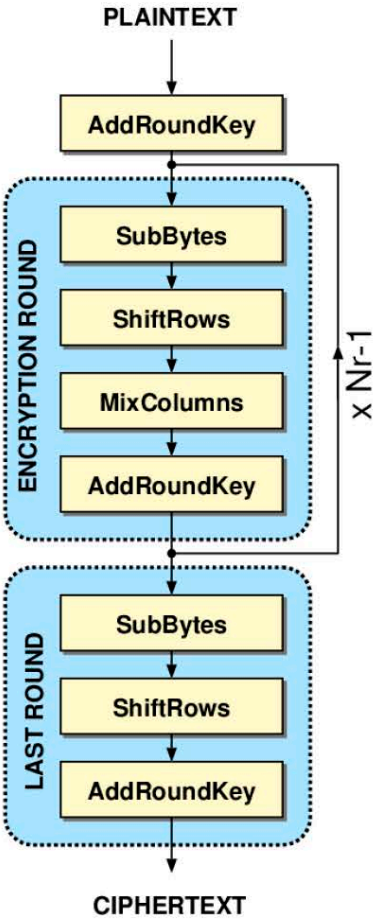


Рисунок 1.12. Реалізація шифрування за алгоритмом AES

Для кожної ітерації даного методу обирається свій підключ. Функції, що застосовуються у даному алгоритмі:

- SubBytes – таблична заміна кожного байту. На відміну від DES таблиця відповідностей не змінюється;
- ShiftRows – циклічний зсув вліво;
- MixColumns – операції поліноміальної арифметики або дії над полем Галуа $GF(2^8)$;
- AddRoundKey – побітова операція XOR з відповідним байтом ключа ітерації.

Перевагою використання даного методу є швидкість, оскільки він оперує байтами та виконання відбувається на процесорі. Зазвичай на цей алгоритм здійснюються атаки сторонніми каналами. Недоліками є:

- однакове шифрування для ідентичних даних (кожний блок завжди кодується однаково);
- проста структура;
- важка імплементація за допомогою програмного забезпечення.

1.2.4 Алгоритм RC4

Алгоритм RC4 є потоковим шифром, який генерує псевдовипадковий потік бітів для ключа та часто використовується завдяки простоті реалізації та швидкості роботи, кодуючи великі потоки даних [6].

Він використовує змінні розміри ключів від 40 до 2048 біт. Вхідні дані – масив байт, ключ – масив бітів. Алгоритм використовує операцію XOR, якщо довжина ключа співпадає з довжиною вхідних даних (рис. 1.13).

Передача таких потоків не виправдана, тому при формуванні кінцевого набору на вхід подається згенерований ключ та застосовуються псевдовипадкові числа на його основі, тобто відбувається розширення даних. Для їх генерації застосовується внутрішній стан (рис. 1.14):

- перестановка 256 байтів (S-блоки підстановок);
- 8-бітові індекси.

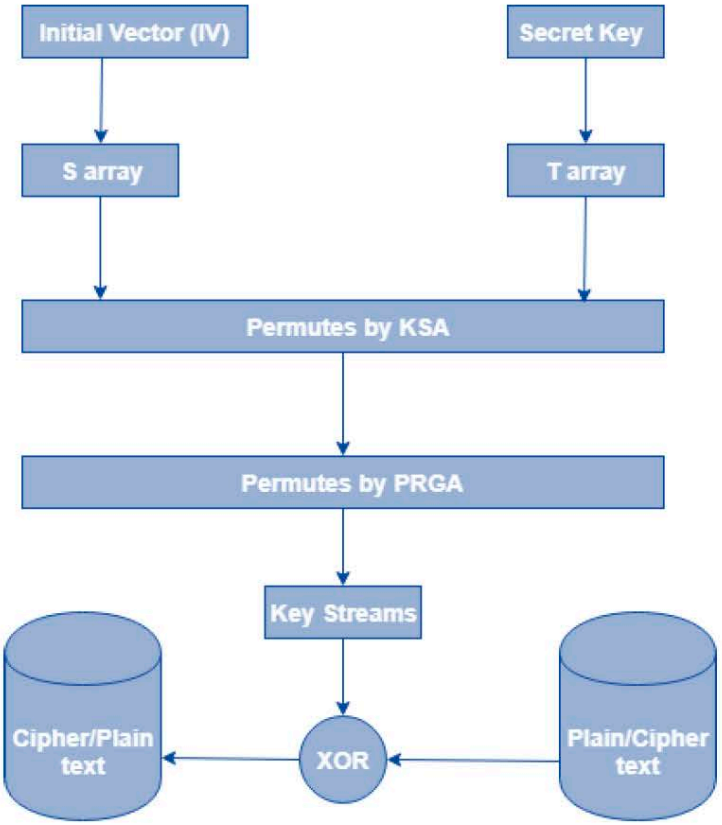


Рисунок 1.13. Реалізація шифрування за алгоритмом RC4

Алгоритм RC4 має недоліки, адже він не проводить автентифікацію та при застосуванні слабких MAC-підписів зламується за допомогою бітової атаки. Також у цьому алгоритмі не можна застосовувати зв'язок між згенерованими ключами, оскільки це не відповідає принципам Шеннона. Алгоритм RC4-drop[n] є модифікованою версією цього алгоритму.

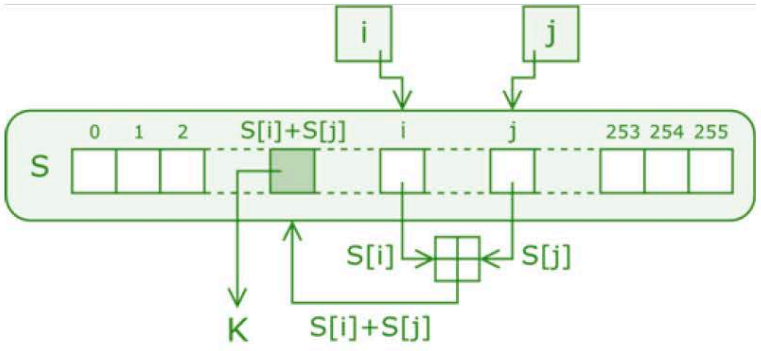


Рисунок 1.14. Генерування псевдовипадкового ключу

1.2.5 Визначення найкращих потових алгоритмів шифрування

Враховуючи зазначені вище недоліки існуючих алгоритмів був створений проект для пошуку найкращих потокових алгоритмів шифрування – «eSTREAM». До нього увійшли такі алгоритми шифрування:

– Sosemanuk – на основі SNOW 2.0, який застосовує ключ довжиною 128 та 256 біт. Переваги даного алгоритму – швидкість та стійкість до всіх існуючих на сьогодні атаках. При застосуванні обчислювальних ресурсів 2^{138} була проведена найуспішніша атака;

– Salsa 20/12 – використовує хеш-функцію з 12 циклами. Має високу швидкодію завдяки незалежному перетворенню стовпців та рядків, перевершуючи при цьому інші шифросистеми: AES та RC4. Поки не було повідомлень про атаки на даний вид Salsa20;

– ChaCha є спорідненим сімейством потокових шифрів, метою якого було поліпшення перемішування даних за один раунд і поліпшення криптостійкості при тій же, або навіть трохи більшій швидкості. Основний блок системи тут працює інакше. Тепер кожна операція змінює одне з слів. Зміни відбуваються циклічно «у зворотний бік», починаючи з 0-го слова. Чергуються операції додавання побітової суми разом із зсувом, кожне слово складається з попереднім. Тут використовуються ті ж самі арифметичні операції, але кожне слово змінюється два рази за перетворення замість одного;

– Rabbit – генерує бітовий потік та застосовує перемішування з використанням арифметичних операцій внутрішніх станів між ітераціями. Недолік – забезпечує захист лише для ключа довжиною 128 біт;

– HC-128 – використовує дві секретні таблиці, які складаються з 512 32-бітних елементів. Після їх застосування на кожному кроці, один регістр змінюється за допомогою нелінійного зворотного зв'язку. Функції та виходи даних нелінійні, тому швидкий злом даного алгоритму неможливий;

– Grain – складається з 80-розрядного LFSR та відповідного NLFSR. Виконує 16 раундів паралельно. Проте, в 2006 була здійснена атака грубої сили, а також були знайдені зв'язані ключі та початкове значення для даного шифру;

– MICKEY – використовує такти зсувних регістрів та покращені методи генерації псевдовипадкових чисел, застосовується в апаратних платформах з обмеженими ресурсами;

– Trivium – використовує 80-розрядний ключ та відповідний вектор

ініціалізації. Реалізований за допомогою трьох зсувних регістрів – 93, 84 та 111 біт. Шифрування використовує операцію XOR для кожного члену тексту та ключового потоку Z. Найшвидший спосіб атак – пошук ключів [7].

Порівняння швидкостей використання поширеного алгоритму RC4 та алгоритмів з проекту «eSTREAM» дозволяє зробити висновок, що RC4 повільніший, ніш більш просунуті алгоритми, наведені вище, оскільки не використовує переваг процесорів, а лише виконує бітові операції [8].

1.3 Аналіз технічного завдання і постановка задачі проектування

Основні проблеми у створенні криптографічних алгоритмів, які використовують зловмисники при атаках, – генерація випадкових чисел, ненадійно захищена передача ключів, фізичні властивості проектування обчислювальних систем, критерії підтвердження актуальності та повноти даних. Криптографічні системи можна зламати, якщо правильно застосовувати криптоаналіз та знання математики. Основні стратегії кібератак:

- вичерпний пошук (грубої сили) – перевірка всіх можливих ключів;
- частотний аналіз – знання розподілу послідовностей символів, які зберігаються у відкритому повідомленні та шифротексті;
- атака «людина посередині» – зловмисники перехоплюють трафік, зазвичай після запиту на надсилання відкритого ключа, а згодом можуть модифікувати чи підслухати сторони;
- атака сторонніми каналами – знання фізичних слабких місць реалізацій криптосистем;
- диференціальний аналіз несправностей – вивчення помилок, які виникли у криптосистемі під час атак.

Криптографія вирішує проблеми обмеження доступу до даних, а не проблеми з безпекою. Якщо необхідно захистити дані, треба використовувати рекомендації затверджені NIST або FIPS. Алгоритми, що пройшли сертифікацію, постійно тестуються, проходять аналіз та покращуються.

Задачею даного дипломного проекту є визначення вразливостей популярних

алгоритмів шифрування, вибір оптимального алгоритму та розробка моделі кодеку на базі алгоритму симетричного шифрування. При цьому необхідно обрати достатньо криптостійку систему, яка б забезпечувала можливість захисту конфіденційної інформації від третіх осіб. У якості такої системи, відповідно до виконаного вище огляду та технічного завдання, обрано алгоритм ChaCha – один з ефективніших найкращому на сьогоднішній день алгоритмів у захисті даних в випадках зберігання на персональному комп'ютері. Створюваний кодек має виконувати шифрування і дешифрування введених користувачем текстових рядків за допомогою алгоритму ChaCha. Розроблений кодек буде відрізнятися від базового алгоритму ChaCha тим, що буде використовувати швидку генерацію ключа, криптостійку псевдовипадкову генерацію початкового значення для різного шифрування однакових даних. Для перевірки інформації на цілісність буде створено хеш, який буде використовувати необоротну функцію для унеможливлення декодування. Для створення додатку з візуальним інтерфейсом для тестування цього алгоритму шифрування буде застосовано платформу QT [9].

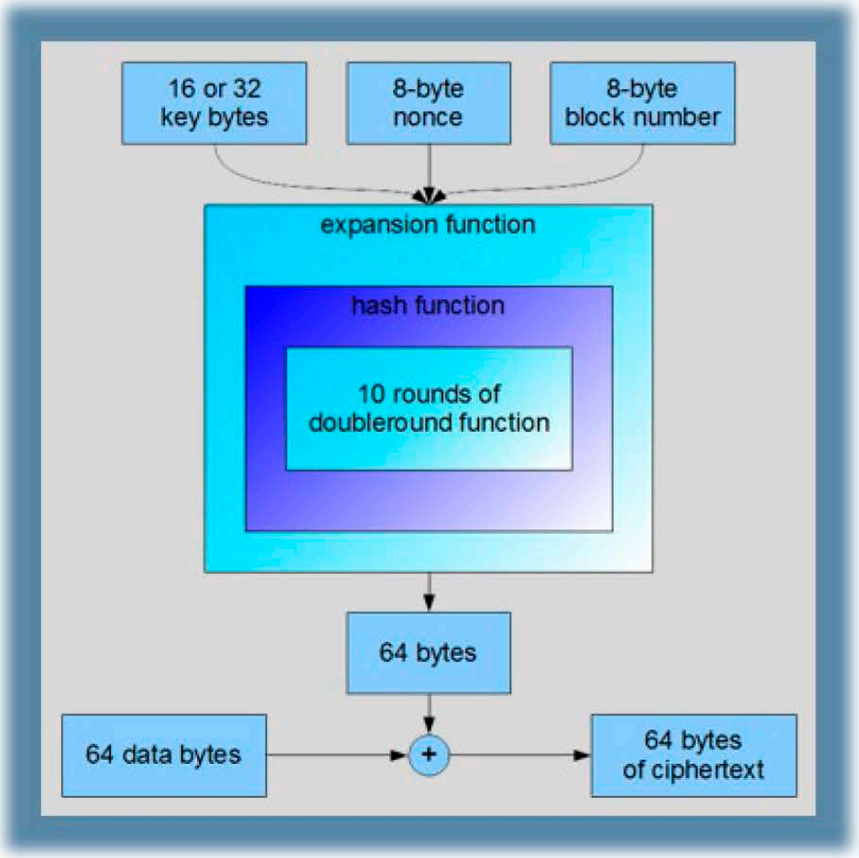
1.4 Розробка моделі кодеку та його програмна реалізація

1.4.1 Застосування алгоритму шифрування ChaCha

Алгоритм ChaCha буде застосовано для шифрування повідомлення з формуванням ключа за допомогою вихорю Мерсенна та реалізацією ініціалізаційного вектору із застосуванням криптостійкого алгоритму Блум-Блум-Шуба. Для перевірки на цілісність даних буде використовуватись алгоритм хешування SHA-512.

Особливістю потокового методу є те, що кодування та декодування виконуються однією й тою ж самою функцією. Швидкість алгоритму ChaCha більше, ніж у RC4 за рахунок того, що виконання кожної ітерації здійснюється за скінчений час, що дає можливість захиститися від атак по часу (аналіз, скільки система витрачає для завершення роботи криптографічних алгоритмів для різного набору шифротекстів чи ключів). Алгоритм має добрі показники для застосуванні у програмному та апаратному забезпеченні. Для процесорів з архітектурою x86

реалізувати метод досить просто, оскільки ця система містить увесь набір команд (інструкцій SSE2), необхідний для роботи даного алгоритму. Для реалізації даного алгоритму варто створювати ключ (KEY) довжиною 128 біт (16 байт) або 256 біт (32 байти) та вектору ініціалізації (IV) – 64 біти (8 байт). В основі ж лежить хеш-функція 64 байти, яка працює разом з лічильником і становить 20 циклів, які виконуються над внутрішнім станом. Nonce має період зміни значення від 0 до $2^{64} - 1$ (рис.1.15).



SALSA20 Algorithm Block Diagram

Рисунок 1.15. Блок-схема алгоритму, на якому заснований кодек ChaCha

У алгоритмі ChaCha застосовуються три прості операції: виключне або (побітове додавання), бітовий циклічний зсув вліво та операція додавання (рис. 1.16). Функція quarterround(a, b, c, d), де a, b, c, d-слова, в ChaCha виглядає наступним чином:

$$\begin{aligned} a+ &= b;\ \ d\oplus = a;\ \ d \lll= 16;\\ c+ &= d;\ \ b\oplus = c;\ \ b \lll= 12;\\ a+ &= b;\ \ d\oplus = a;\ \ d \lll= 8;\\ c+ &= d;\ \ b\oplus = c;\ \ b \lll= 7; \end{aligned} \qquad (1.13)$$

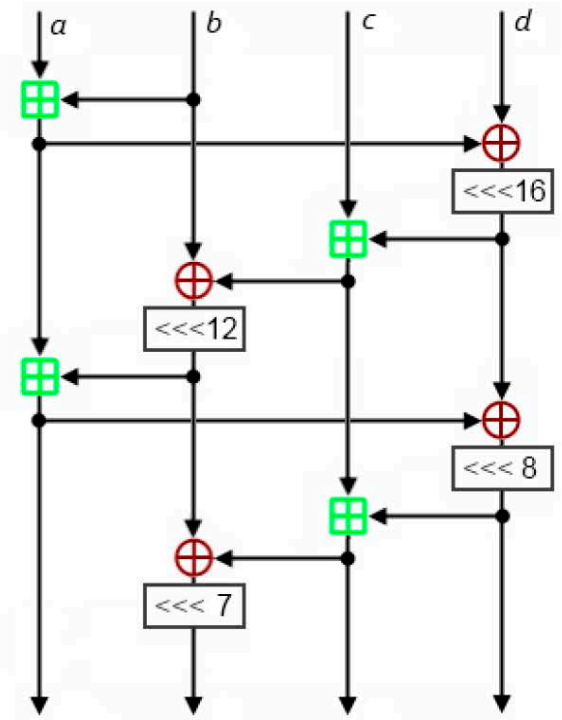


Рисунок 1.16. Цикл роботи алгоритму ChaCha функції quarterround(a, b, c, d)

У алгоритмі ChaCha можна передати розмір даних, які будуть кодуватися. Недоліком ChaCha є те, що її можна використовувати лише для захисту особистих даних, які зберігаються на персональному комп'ютері або на жорсткому диску. Оскільки цілісність зашифрованого тексту не перевіряється, для більш важливих даних у комерційних цілях, при роботі з банками та іншому, необхідне автентифіковане шифрування [10].

1.4.2 Застосування вихорю Мерсенна для генерації випадкових чисел

Застосування вихорю Мерсенна дозволяє реалізувати достатньо криптостійкий генератор псевдовипадкових чисел з періодом повторення деякого числа $2^{19937} - 1$, що працює ефективніше приблизно у двадцять разів відносно апаратного RDRand. Дане значення періоду дозволяє уникнути атаки зловмисників з урахуванням, що вони будуть намагатися передбачувати наступні значення. Алгоритм генерує 32-бітні значення, які не мають залежності від попередніх, що забезпечує деякий рівень непередбачуваності (рис. 1.17). Вихор Мерсенна виконує певні перетворення, що створюють малу кореляцію між послідовними значеннями та забезпечують наближену послідовність з рівномірним розподілом. Натуральне число Мерсенна (M_n) можна обчислити за формулою (1.14).

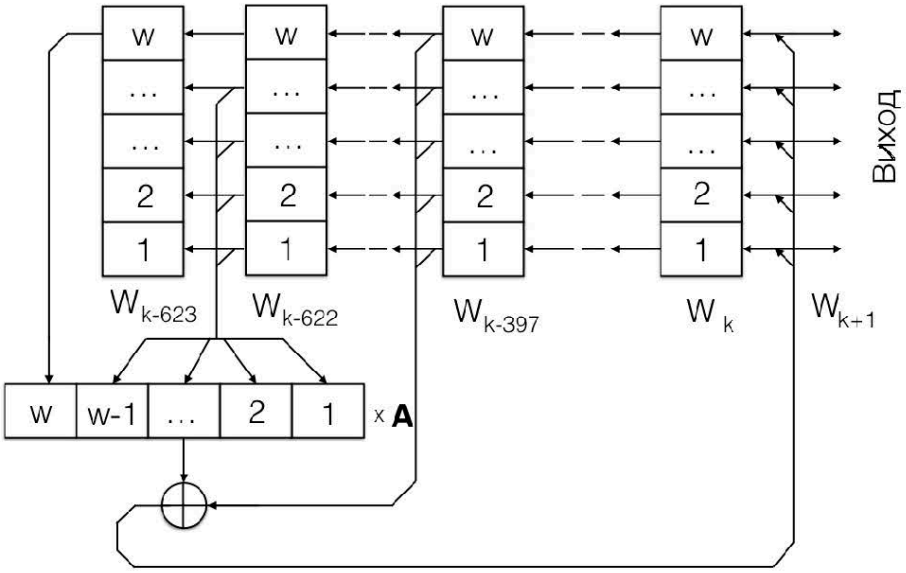


Рисунок 1.17. Спрощена схема вихорю Мерсенна

$$Mn = 2^n - 1 \qquad (1.14)$$

де n – натуральне число.

На вхід схеми має подаватися кілька мільйонів 32-розрядних цілих чисел. На виході має бути отримано значення, яке рівномірно розподілене між 0 та 1. При цьому враховуються наступні особливості:

– відстані між випадково вибраними точками мають мати Пуассонівський розподіл;

– обирається випадкове число на діапазоні [0, 1) та множиться на 2^{31}, поки не буде отримано 1;

– обчислюються ранги матриць, сформованих з деякої кількості випадкових чисел для матриці {0,1};

– генерується послідовність чисел на діапазоні [0, 1), де додаються 100 послідовних чисел і суми мають бути нормально розподілені з певною дисперсією;

– перестановки з п'яти чисел мають бути рівномірно розподілені.

Для використання даного алгоритму у криптостійких системах необхідно додатково використовувати алгоритми хешування. Метод застосовує рекурсивну частину, яка реалізується за допомогою 624 елементів регістру зсуву з лінійним зворотнім зв'язком та загартування, що підсилює рівномірність розподілу на великих послідовностях бітових векторів. На вхід функції ініціалізації подається

певне початкове значення – seed, за допомогою якого заповнюється весь регістр. На кожному 624 кроці йде перемішування внутрішнього стану. Недоліком даного алгоритму є те, що якщо у зловмисника є задані 624 числа, які згенеровані цим методом, він може відновити внутрішній стан регістрів та передбачати значення з стовідсотковою ймовірністю.

Даний алгоритм був реалізований за допомогою бібліотеки <random> у просторі імен std у проекті mt19937. Проініціалізований він був числом, згенерованим за допомогою спеціального об'єкту seed_seq на основні генерації random_device (true-random-number), значення якого розподіляються рівномірно по 32-бітному діапазону. Це дає змогу для генератора, який вимагає велику ентропію, дати початкову константу [11].

1.4.3 Застосування алгоритму BBS

У даному проекті алгоритм BBS (Блум-Блум-Шуба) використано для генерації вектору ініціалізації, оскільки від нього залежить, як будуть шифруватися однакові повідомлення. На відміну від вихору Мерсенна, даний метод набагато повільніший, але надійніший, що забезпечується створенням значень шляхом факторизації:

$$x_{n+1} = x_n^2 \ mod \ (p * q) \qquad (1.14)$$

де p, q – великі прості числа; x_{n+1} – початкове значення для кожної наступної ітерації; x_n – вихідні дані.

Алгоритм Блум-Блум-Шуба формує вихідні дані з кожним кроком шляхом взяття найменш значущих бітів або біта парності. Спочатку генеруються прості числа, які перемножують, потім відшукується велике взаємно-просте число з отриманим результатом. Застосовується формула (1.14) та береться останній біт значення x_n (рис.1.18).

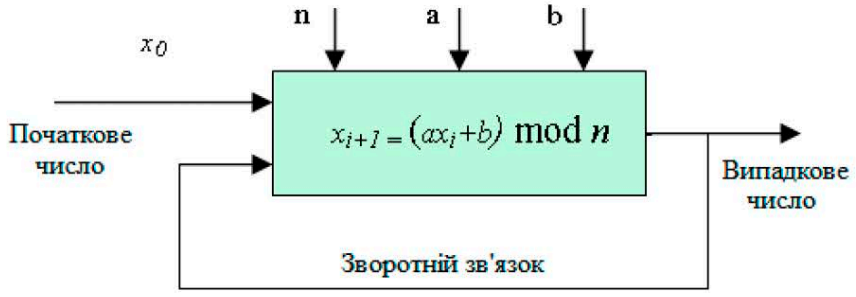


Рисунок 1.18. Схема реалізації алгоритму BBS

1.4.4 Застосування алгоритму хешування SHA-512

SHA-512 є безпечним алгоритмом, що використовується для створення хеш-суми (дайджесту) вхідних даних з метою встановлення цілісності повідомлення чи ідентифікації. Цей алгоритм базується на хеш-функції Меркла-Демґарда (рис. 1.19).

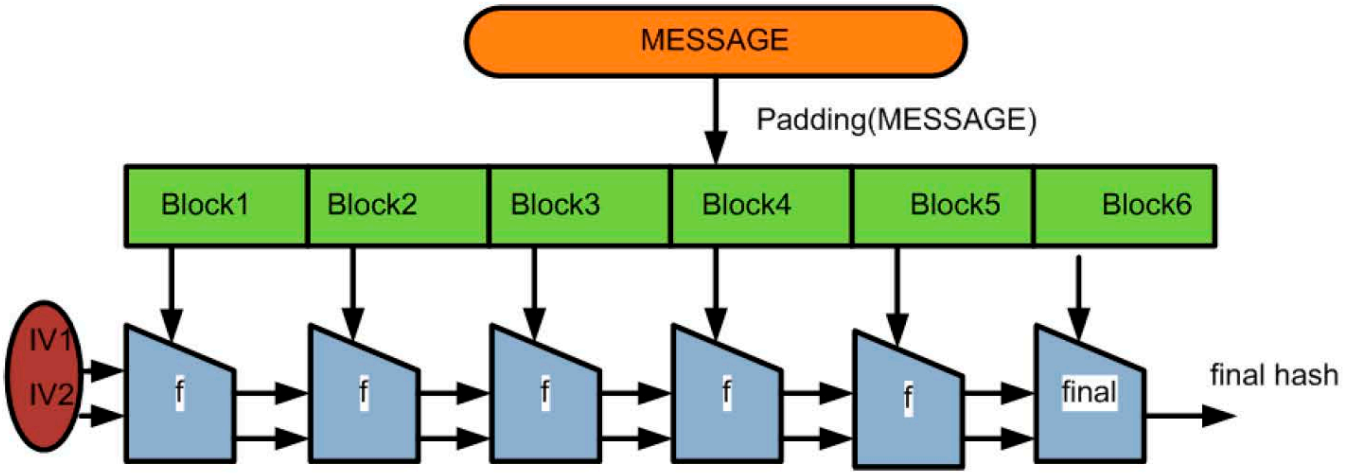


Рисунок 1.19. Схема реалізації алгоритму Меркла-Демґарда для хешування

Спочатку функція застосовує доповнення для створення блоків, які кратні 512, розбиває вхідні дані і до кожного блоку застосовує алгоритм стискання.

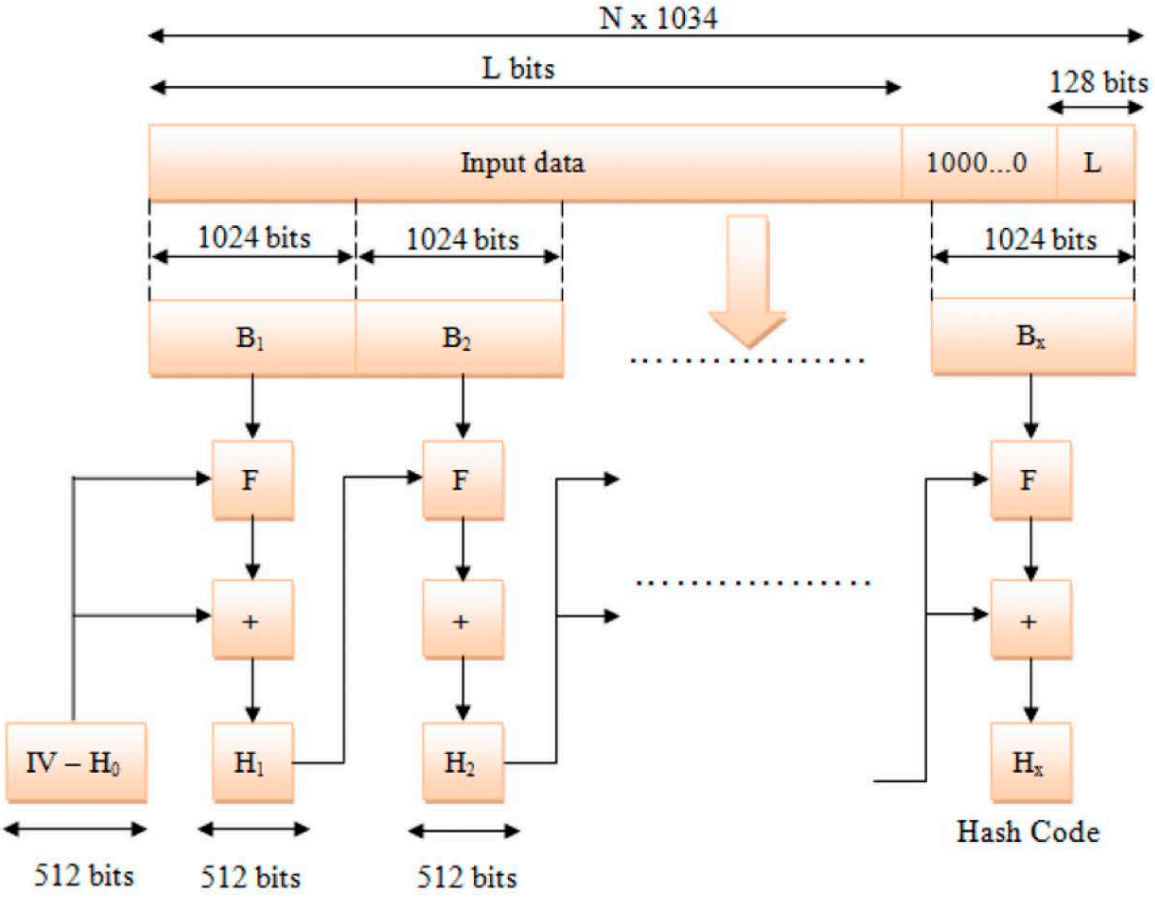


Рисунок 1.20. Структура алгоритму SHA-512

Алгоритм SHA-512 використовує ініціалізаційний вектор, завершальну функцію (застосовує стискання, змішування або ж лавиновий ефект) і стискання (поєднується з наступним блоком повідомлення), створює 1024-бітні блоки і здатен приймати довжину рядка 2^{128} біт. Після його застосування 80 раундів видають 64 байти хешу, початкова інформація інтерпретується у вигляді 128 шістнадцяткових цифр (рис. 1.20). У раундах застосовуються константи – перші 80 простих чисел. Проте, навіть такі функції піддаються атакам, через відшукання колізій. Особливістю хеш-функцій є те, що вони не оборотні і використовуються для електронного цифрового підпису, криптовалюти, зберігання паролів. Таке сімейство, як SHA-2 вразливе до атак з розширенням довжини, але на сьогодні, не має жодної інформації щодо зламу версії SHA-512 [12].

1.4.5 Вибір програмних засобів розробки

Програмна реалізація проекту буде виконуватись з використанням об'єктно-орієнтованої мови програмування C++. При цьому будуть використовуватись засоби мови C++для роботи з пам'яттю на низькому рівні. Для створення графічного інтерфейсу користувача (GUI) буде застосовані середовище розробки QT. Даний вибір забезпечує можливість використовувати одну мову програмування та фреймворк без додаткових інструментів. C++ є мовою, якою можна писати швидкі, прості, а головне компактні програми. Оскільки мова низькорівнева, нею можна писати різні програми – від драйверів до ігор. Ця мова надає розробникам повний контроль над керуванням пам'яттю, що підходить для концепції реалізації алгоритму шифрування ChaCha, що передбачає бітові операції.

Для шифрування даних буде застосовано бібліотеку Crypto++, яка містить криптографічні класи з реалізацією різних алгоритмів шифрування та псевдогенераторів чисел.

Інтегроване середовище розробки QT (рис.1.21) є IDE-платформою, що підтримує компілятори GCC, G++, Visual Studio, дає змогу відлагоджувати код, може застосовуватися на будь-якій операційній системі завдяки кросплатформності, має добре документовану структуру, містить засоби об'єктно-орієнтованого програмування, що дозволяє застосовувати наслідування.

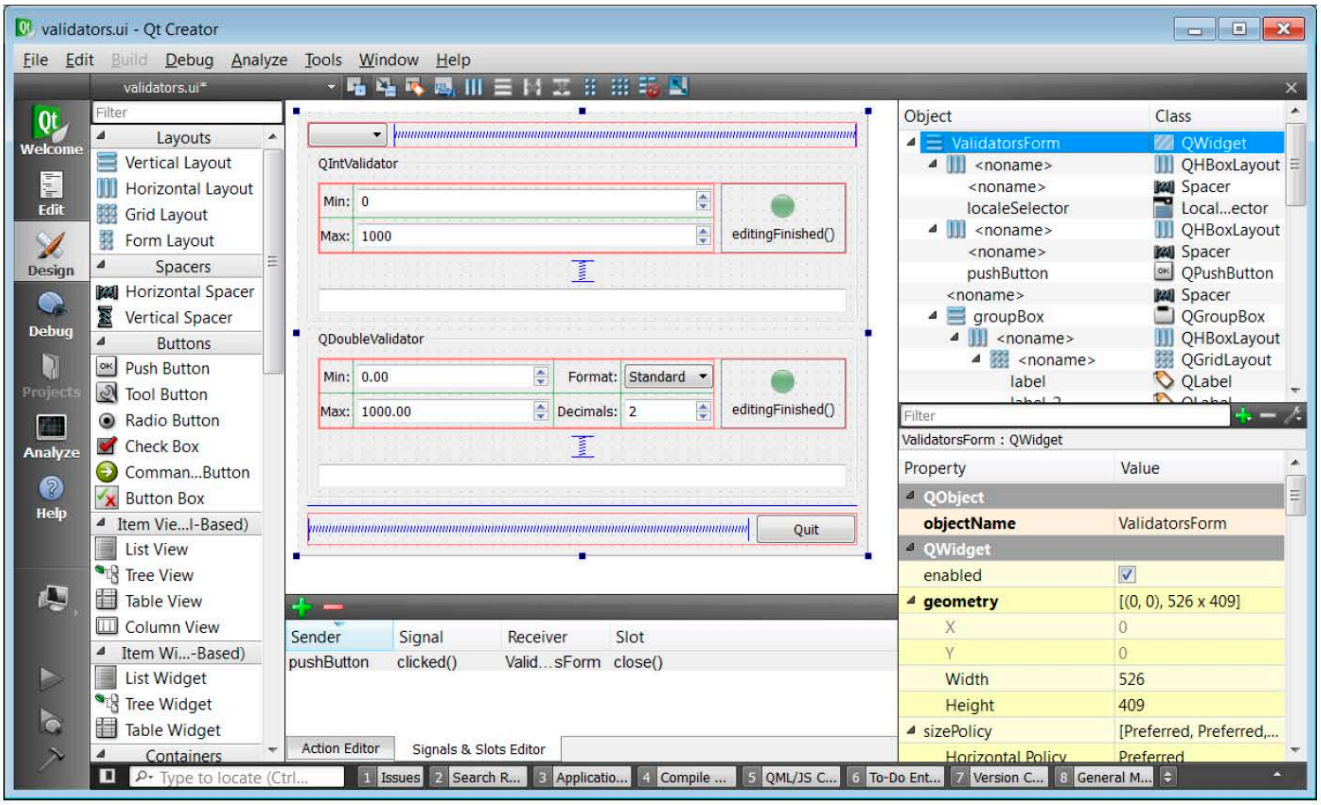


Рисунок 1.21. Інтерфейс IDE QT Creator

Віджети у IDE QT Creator надсилають сигнали (слоти), що містять інформацію про подію для зв'язку між реалізацією інтерфейсу для користувача та коду. Для автоматизації генерації Makefiles можуть використовуватись віджети cmake, qmake. Для контролю версій програми буде використано GIT, що дозволить візуально прослідкувати зміни та швидке злиття гілок. GIT працює з даними за допомогою снапшотів – у певний момент часу він робить копію (знімок) файлової системи та зберігає посилання на них. Уся історія проекту зберігається локально на диску пристрою, тому швидкість роботи даної системи миттєва [13].

1.4.6 Вибір програмних складових програмного проекту

Створюване програмне забезпечення буде складатися з .ui-файлів (user interface – реалізація візуальної частини), .h-файлів (заголовні файли – декларація змінних, класів та функцій) та .cpp-файлів (вихідний код на мові C++). Для роботи з байтами буде використано стандартну бібліотеку C++ cstdint та QT QByteArray. У проекті будуть застосовані такі модулі:

– mainwindow.ui, mainwindow.cpp, mainwindow.h – реалізація головного вікна для вибору та виконання кодування даних, відкриття вікна з ключем, виходу з

програми;

– chacha.cpp, chacha.h – реалізація алгоритму Chacha;

– key.ui, key.cpp, key.h – створення та виведення ключа та вектору ініціалізації на вікно «Ключ», які реалізуються за допомогою вихорю Мерсенна, реалізованого у стандартній бібліотеці random та з використанням алгоритму BBS;

– constants.h – зберігає підключення основних бібліотек, та констант;

– crypto.cpp, crypto.h – генерація хешу з використанням SHA-512, реалізація кодування та декодування даних з використанням функцій з набору файлів Chacha.

Для подальшого запуску програми на персональному комп'ютері необхідне програмне забезпечення інтегрованого середовища розробки QT з стандартними бібліотеками. Відповідні заголовні файли треба додати у папку з розширенням .exe: Qt5Core.dll, Qt5Gui.dll, Qt5Widgets.dll, libgcc_s_seh-1.dll, libstdc++-6.dll, qt.conf. До папок \plugins\platforms треба додати файли qwindows.dll та qminimal.dll.

1.4.7 Моделювання роботи застосованих алгоритмів

Для моделювання роботи застосованих алгоритмів SHA-512 та ChaCha використовувалось програма CrypTool2. На рис.1.22 наведено результат моделювання за алгоритмом SHA-512 по отриманню хеш-суми рядку "Доброго дня, шановна комісія. Мене звуть Тимофєєва Валерія". В результаті отримано хеш-суму "B8 E7 26 2E F3 42 6D 95 3D 52 AC F5 9F E5 EC E8 DF 00 60 82 10 5F 5F 9E 1D 29 FC 4A 52 2E 1A 06".

На рис.1.23-1.24 наведені результати моделювання за алгоритмом ChaCha для відкритого тексту "Доброго дня, шановна комісія. Мене звуть Тимофєєва Валерія". В результаті отримано шифротекст "9B 98 23 FD 1C 07 E5 2C 8B 20 68 CD 43 54 17 A1 A0 FB 6D 2C 45 AF 3F 64 43 C0 4C F4 0F 64 48 BD B7 B6 58 8E E6 A9 FA 06 03 42 E6 00 73 2F BE 46 B7 AF 47 CB 84 88 8E 3F 3A 1C E2 48 66 06 EC D7 41 1B 4C F5 0C 7F 14 64 33 68 4B C4 EA AC 3D CB 94 5B 7E 25 6A 6A 2B 40 76 74 23 B3 6C 97 6B 5A 8C 48 B5 5D 98 FD 12 BB 8B B5 11".

На рис.1.25 наведені результати шифрування і дешифрування за алгоритмом ChaCha для відкритого тексту "Доброго дня, шановна комісія. Мене звуть Тимофєєва Валерія".

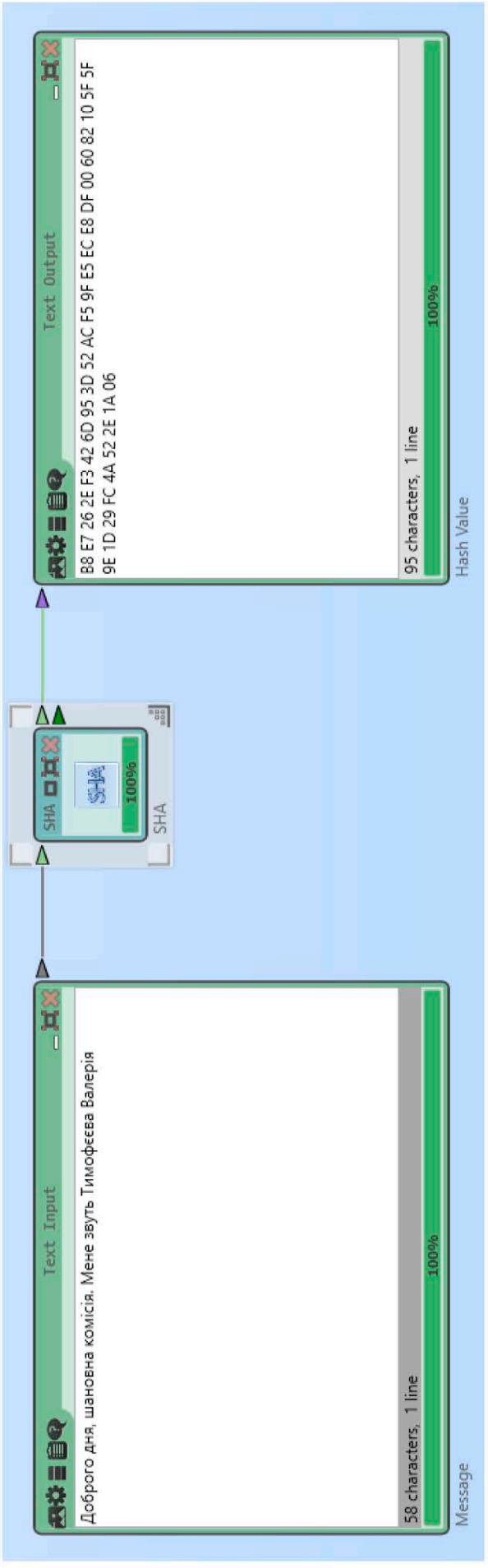


Рисунок 1.22. Результат моделювання за алгоритмом SHA-512

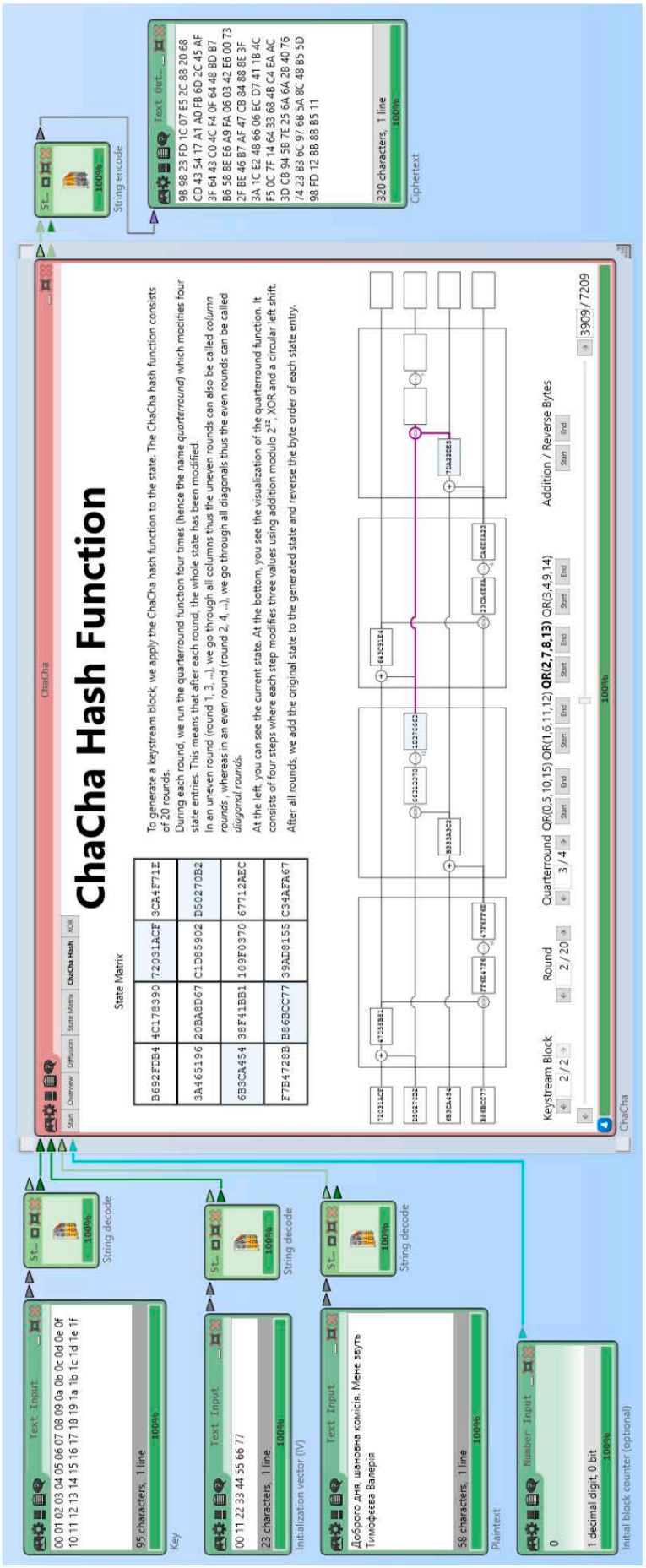


Рисунок 1.23. Результат моделювання шифрування
за алгоритмом ChaCha на кроці 3909

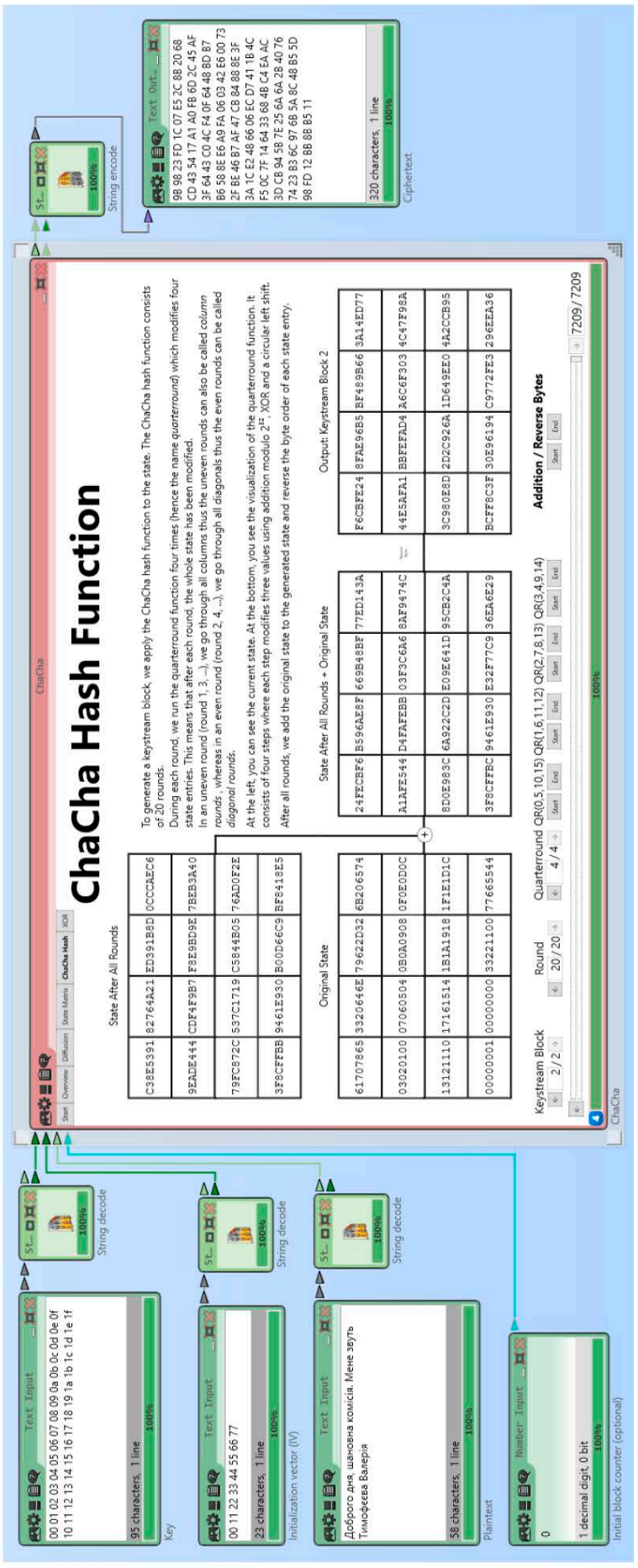


Рисунок 1.24. Результат моделювання шифрування
за алгоритмом ChaCha на кроці 7209

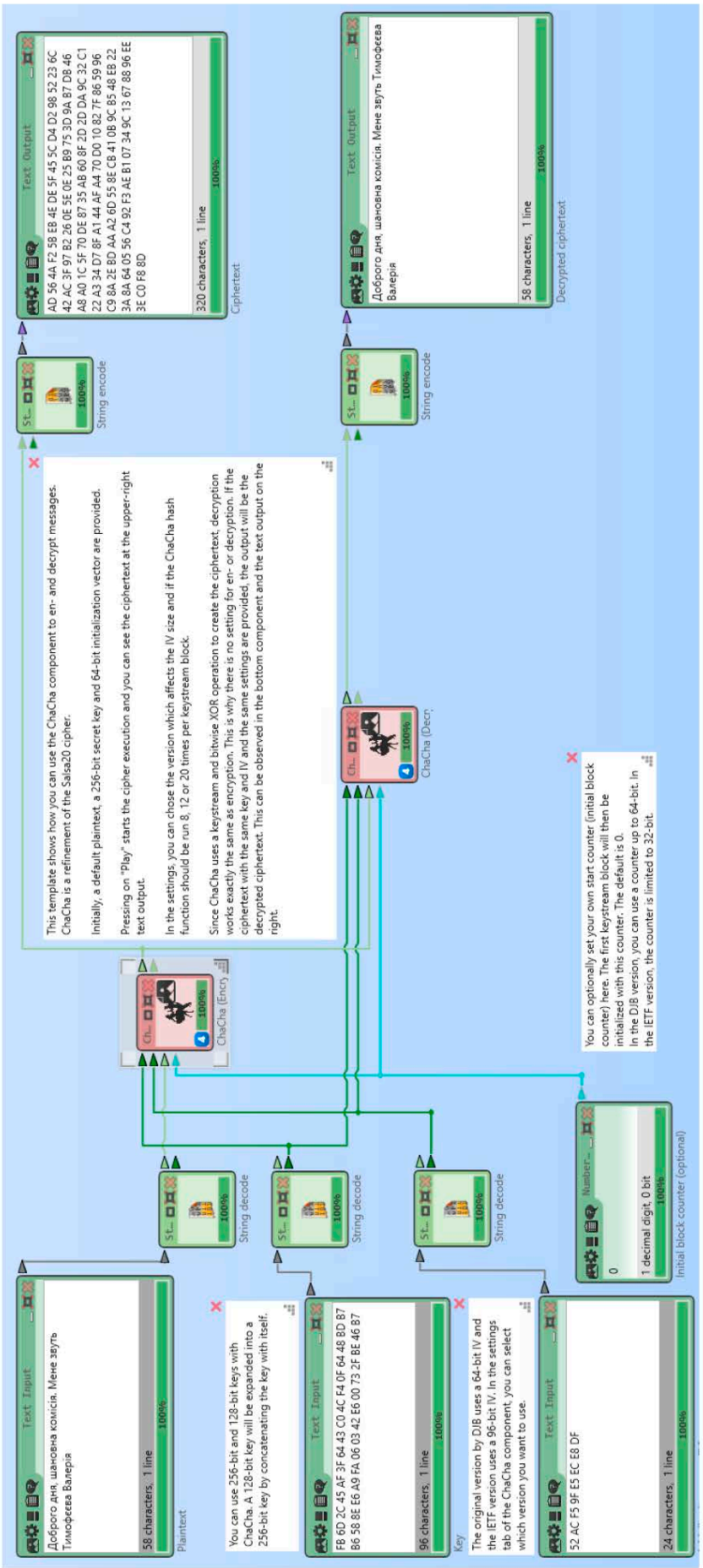


Рисунок 1.25. Результат моделювання шифрування та дешифрування за алгоритмом ChaCha

В результаті отримано шифротекст “AD 56 4A F2 5B EB 4E DE 5F 45 5C D4 D2 98 52 23 6C 42 AC 3F 97 B2 26 0E 5E 0E 25 B9 75 3D 9A B7 DB 46 A8 A0 1C 5F 70 DE 87 35 AB 60 8F 2D 2D DA 9C 32 C1 22 A3 34 D7 8F A1 44 AF A4 70 D0 10 82 7F 86 59 96 C9 8A 2E BD AA A2 6D 55 8E CB 41 0B 9C B5 48 EB 22 3A 8A 64 05 56 C4 92 F3 AE B1 07 34 9C 13 67 88 96 EE 3E C0 F8 8D”. Ключем при цьому є послідовність “FB 6D 2C 45 AF 3F 64 43 C0 4C F4 0F 64 48 BD B7 B6 58 8E E6 A9 FA 06 03 42 E6 00 73 2F BE 46 B7”, а вектор ініціалізації дорівнює значенню “52 AC F5 9F E5 EC E8 DF”.

1.4.8 Розробка програмного коду та створення інтерфейсу

Текст основної функції шифрування за алгоритмом ChaCha розробленої програми мовою C++ наведено у Додатку А. Розробка виконано у інтегрованому середовищі розробки QT під керуванням операційної системи Windows 10 Pro x64. Інтерфейс головного вікна розробленої програми-кодеку наведено на рис. 1.26.

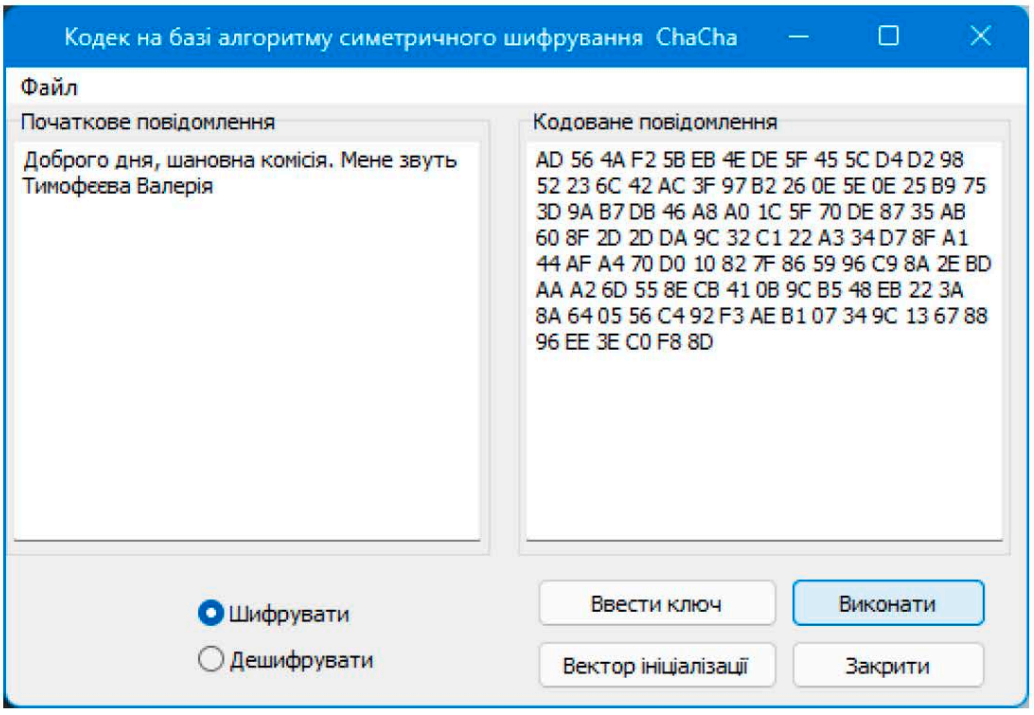


Рисунок 1.26. Головне вікно програми-кодеку на базі алгоритму ChaCha

Для шифрування початкового повідомлення необхідно у поле «Початкове повідомлення» ввести текст та вибрати опцію «Шифрувати». При цьому спочатку необхідно натиснути на кнопки «Ввести ключ» та «Вектор ініціалізації», після чого відкриються вікна генерування ключу та вектору ініціалізації (рис. 1.27).

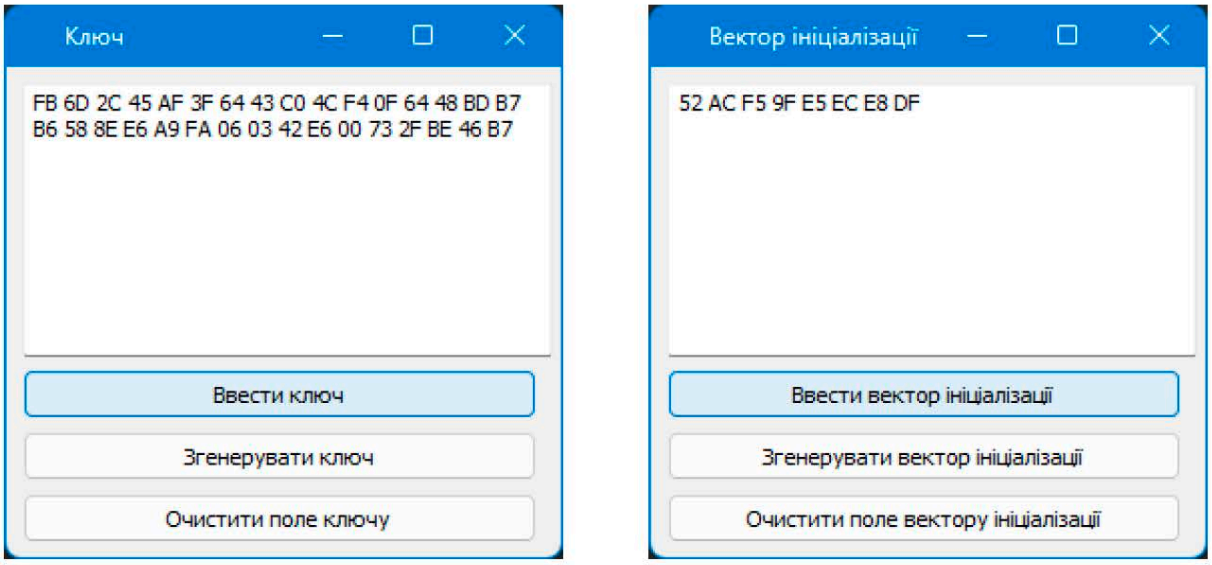


Рисунок 1.27. Генерування або введення ключу та вектору ініціалізації

Після закриття вікна генерування ключу відбувається перехід до основного вікна. При натисненні кнопки «Виконати» шифрований або дешифрований за допомогою ChaCha текст з'явиться у полі «Кодоване повідомлення» (рис. 1.28).

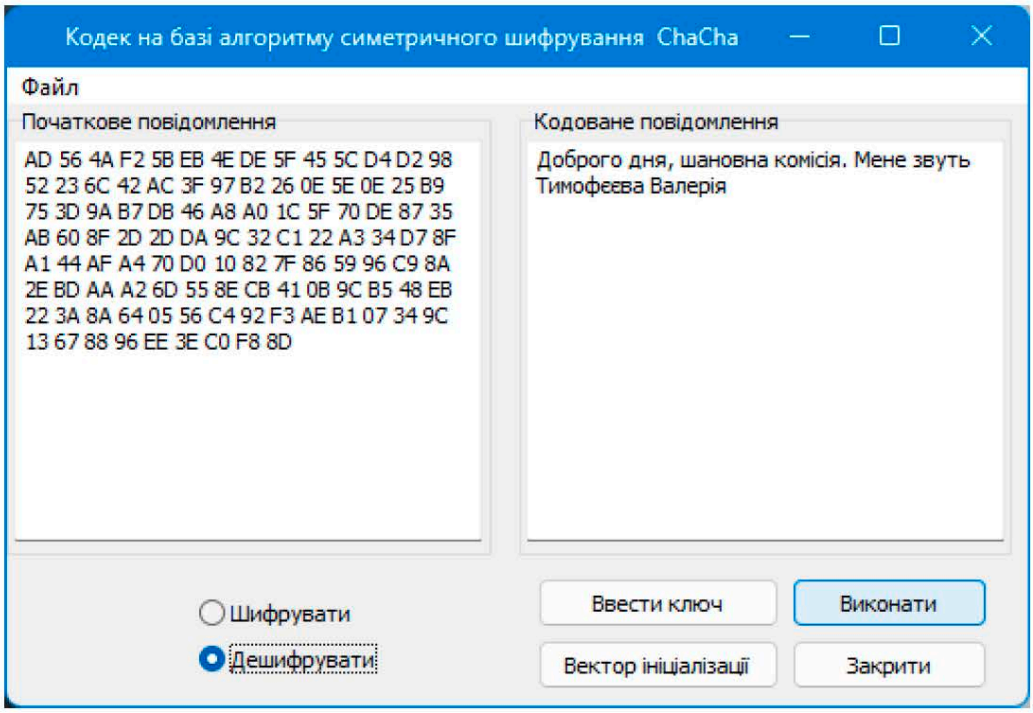


Рисунок 1.28. Приклад дешифрування програмою-кодеком повідомлення

При відсутності згенерованого ключу / вектору ініціалізації шифрування дешифрування повідомлень буде невдалим, в результаті чого передбачене виведення повідомлення про помилку (рис. 1.29).

За наведеними вище скріншотами (рис.1.26, рис.1.28) видно, що

зашифровані/дешифровані розробленим кодеком повідомлення повністю співпадають з результатами модулювання у програмі Cryptool2, наведеними на рис.1.25, що свідчить про правильну роботу розробленого додатку.

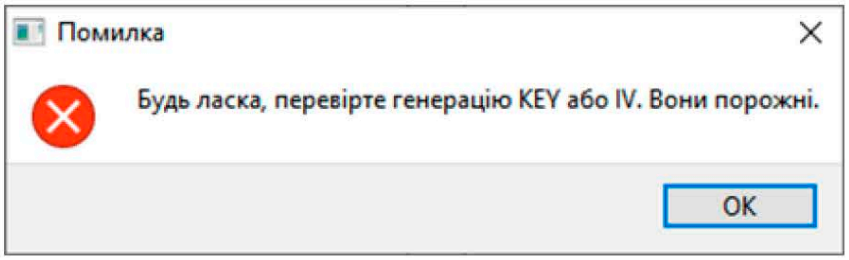


Рисунок 1.29. Повідомлення про відсутність ключу / вектору ініціалізації

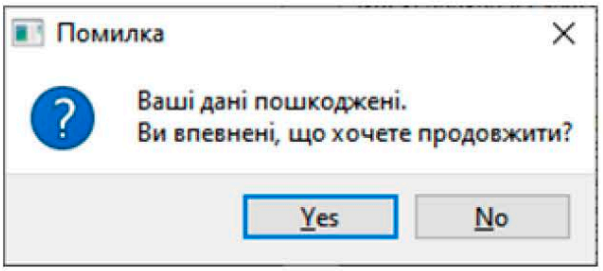


Рисунок 1.30. Повідомлення про порушення цілісності даних

Для перевірки цілісності даних генерується хеш з вихідних даних та порівнюється з тим, який був згенерований до цього. У випадку, коли вони не співпадають, система видає помилку про те, що дані були змінені (рис. 1.30). При натисканні «Yes» програма буде намагатися декодувати пошкоджені дані, що призведе до неправильних вихідних даних.

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

Тема даного дипломного проекту «Розробка моделі кодексу на базі алгоритму симетричного шифрування». В роботі виконується розробка моделі кодеку симетричного потокового алгоритму шифрування ChaCha з застосуванням рівномірного розподілу. Упровадження цієї системи дозволить збільшити швидкість обробки даних для шифрування із забезпеченням захисту від зловмисників.

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість ПП визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

Таблиця 2.1. Каталог аналогів

Найменування ПП	**Обсяг функції ПП – V_o, усл. машинних командах.**
1. ПП автоматизованих розрахунків	1300 – 8600
2. ПП загальної математики і ПП імітаційного моделювання	1800 – 8800
3. ПП введення інформації	1060 – 5750

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для

нашого варіанта виділено сірим кольором.

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця.2.2. Норма часу

Обсяг ПП, тис.умов.машинних команд	**Норма часу, люд/год**
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): $T^a = 244 \times 0,7 = 170,08$ (люд/годин).

Трудомісткість програмного продукту визначаємо по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5)

Таблиця 2.3. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.5. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Т^а *L_1*К_н= 170,08*0,12*0,7= 14,35 \text{ (люд/годин)} \quad (2.4)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Т^а *L_2*К_н=170,08*0,11*0,7 = 13,10 \text{ (люд/годин)} \quad (2.5)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Т^а *L_3*К_н*К_т=170,08*0,61*0,7*0,7= 50,81 \text{ (люд/годин)} \quad (2.6)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр), розробка ТП $N_{тп}$=12 (стр), розробка робочого проекту $N_{рп}$=14(стр), пояснювальна записка відповідно $N_{пз}$=22(стр) Розрахунок зведений у таблицю 2.6.

Таблиця 2.6. Розрахунок трудомісткості ПП

Найменування етапів	**Розрахунок, годин**		
1.ТЗ	$T_{Ртз}$=14,35	$T_{кк}$=0,7*$N_{тз}$= 0,7*2=1,4	$T_{нк}$=0,15*$N_{тз}$=0,15*2= 0,3
2.Розробка ТП	$T_{Ртп}$=13,10	$T_{кк}$=0,7*$N_{тп}$=0,7*12=8 ,4	$T_{нк}$=0,15*$N_{тп}$=0,15*12=1,8
3.Розробка РП	$T_{Ррп}$=50,81	$T_{кк}$=0,7*$N_{рп}$=0.7*14= 9,8	$T_{нк}$=0,15*$N_{рп}$=0,15* 14=2,1
4.Розробка ПЗ	$T_{пз}$=1,5*22 *$N_{пз}$=33	$T_{кк}$=0,7*$N_{тз}$=0,7*22=1 5,4	$T_{нк}$=0,15*$N_{пз}$=0,15*22=3,3
Усього, в т.ч.:	ΣT=153,76		
- на розробку	$\Sigma T_{р}$=111,26		
- контроль керівника		$\Sigma T_{кк}$=35	
- нормоконтроль			$\Sigma T_{нк}$=7,5

2.3 Розрахунок ціни програмного продукту

Для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у таблиці 4.6. Відповідно до статті 8 «Закону про Державний бюджет України на 2023» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2023 року - 6700 гривень; мінімальну погодинну тарифну ставку – 40.46 грн.

Таблиця 2.7. Розрахунок основної заробітної плати виконавців

Найменування робіт	**Трудомісткість робіт, години**	**Погодинна тарифна ставка, грн.**	**Розрахунок, грн.**
1.Розробка ПП	111,26	45,00	5006,25
2.Контроль керівника	35	100,00	3500
3.Нормоконт-роль	7,5	80.00	600
Усього	-	-	ΣЗо= 9106,25

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок

зведемо в таблицю 2.8.

Таблиця 2.8. Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	**Тип, модель**	**Кількість**	**Ціна одиниці, грн.**	**Вартість, грн.**
Папір	Лист А4	50	3.0	150,00
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0,1 \times Вм1 =$ 150*0,1= 15,0
Усього				$В_м=В_{мі}+В_{тр\ з}=$ 165,0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.9.

Таблиця 2.9. Розрахунок статей витрат планової собівартості

Стаття витрат	**Значення, грн.**	**Формула розрахунку**
1. Матеріали	165,0	$В_м = 165,0$
2. Основна заробітна плата	9106,25	$З_о = 9106,25$
3.Додаткова заробітна плата	910,63	$Зд = 0,15 \times Зо$ =0,15*9106,25
4.Відрахування до єдиного фонду соціального внеску	2203,71	$Вє.с.в. = 0,22 \times (Зо + Зд)$ =0,22*(9106,25+ 910,63)
5. Накладні витрати	4553,12	$Внак. = 0,3 \times Зо$ =0,5 * 9106,25
6. Повна собівартість	16938,71	$С_{пов}=В_м+З_о+З_д+Вє.с.в.+Внак.=$165,0+ 9106,25+910,63+2203,71+4553,12

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100=(16938,71*10)= 1693,87 \qquad (2.7)$$

Де р – плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_{пов}+П=16938,71+1693,87=18632,58 \qquad (2.8)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 0.2 * Ц_о =0,2*18632,58= 3726,52 \qquad (2.9)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ =18632,58 + 3726,52 = 22359,10 \qquad (2.10)$$

3 ОХОРОНА ПРАЦІ

Широке впровадження комп'ютерної техніки, що дає змогу автоматизувати багато рутинних операцій комп'ютерної обробки інформації, одержати доступ до численних джерел інформації, швидко проводити потрібні розрахунки тощо, підвищує продуктивність праці. Проте активне впровадження у практику персональних комп'ютерів має і негативну сторону – з'являються фактори, які несприятливо впливають на здоров'я працюючої людини.

Згідно темі дипломного проекту робоче місце користувача послуг складається з персонального комп'ютеру з програмним забезпеченням та призначено для розробки моделі кодексу на базі алгоритму симетричного шифрування.

Тому для нього застосовуються звичайні вимоги безпеки праці для користувача персонального комп'ютеру.

3.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК

Показниками гігієнічних вимог є рівень освітлення, температура, вологість, шум, вібрація, токсичність, загазованість, обмежена загальна м'язова активність (гіподинамії), дія електростатичного поля, неіонізуючих та іонізуючих електромагнітних випромінювань.

3.2 Гігієнічні вимоги до виробничого середовища

Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» призначені для запобігання несприятливої дії на працівників шкідливих факторів, які супроводжують роботу з ВДТ

3.2.1 Вимоги до приміщення

Розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Для приміщень, які призначені для роботи з ВДТ, доцільно обрати орієнтацію вікон на північ або північний схід. На вікнах повинні бути жалюзі, що регулюються, або штори, що дають можливість їх

повністю закривати.

Об'ємно-планувальні рішення будівель та приміщень, де експлуатуються відеодисплейні термінали (ВДТ) мають відповідати вимогам ДСанПІН 3.3.2.007-98

Площа на одне робоче місце для програмістів повинна складати не менше 6 кв.м., а об'єм – не менше 20 куб.м. Стіни повинні бути пофарбовані матовою фарбою.

При приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку, психологічного розвантаження тощо

3.2.2 Освітлення

Для освітлення приміщення, у якому працює програміст, використовується змішане освітлення, тобто сполучення природного й штучного освітлення. Для загального освітлення приміщення, де перебуває робоче місце програміста, використовуються газорозрядні лампи типу ЛД. Нормами для даних робіт установлена необхідна освітленість робочого місця ЕН=300 лк (для робіт високої точності, коли найменший розмір об'єкта розрізнення дорівнює 0,3 – 0,5 мм).
Дана норма освітленості в цілому виконана

3.2.3 Шум

У робочих приміщеннях основними джерелами акустичних шумів є шуми ПЕОМ. ЕОМ є також джерелами шумів електромагнітного походження (коливання елементів електромеханічних пристроїв під впливом змінних магнітних полів). Крім того, у даних приміщеннях, виникає структурний шум, тобто шум, випромінюваний поверхнями коливних конструкцій стін, перекриттів, перегородок будинку у звуковому діапазоні частот.. Для усунення або ослаблення несприятливих шумових впливів доцільно ізолювати робочі приміщення, розміщаючи їх у частинах будинку, найбільш вилучених від міського шуму - розташованих у глибині будинку, звернених вікнами у двір. Перевіряти герметичність корпусів комп'ютерів та своєчасно міняти вентилятори охолодження.

3.3 Вимоги до організації робочого місця працівника

Конструкція робочого місця користувача ПК й взаємне розташування всіх його елементів (сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним і психологічним вимогам, а також характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно росту працюючих для підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в у припустимі рамки від 60 до 90 см. Частота мерехтіння екрана fмер=100 Гц, що відповідає умові fмер>70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехтливість екрана від вікон і приладів штучного освітлення.

3.4 Мікроклімат

Параметри мікроклімату, іонного складу повітря, вміст шкідливих речовин на робочих місцях, оснащених ПК повинні відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки допустимих значень мікроклімату та концентрації позитивних та негативних іонів необхідно передбачати установки або прилади зволоження та/або штучної іонізації, кондиціювання повітря. Рівні інфрачервоного випромінювання не повинні перевищувати граничних

Рівні інфрачервоного випромінювання не повинні перевищувати граничних відповідно до ГОСТ 12.1.005. Вміст озону в повітрі робочої зони не повинен перевищувати 0,1мг/м3; вміст оксидів азоту - 5мг/м3; вміст пилу - 4мг/м3.

3.5 Електробезпека

Приміщення де використовуються імпульсні джерела живлення відповідно до ОНТП24-86 і ПУЕ-87 відноситься до класу приміщень без підвищеної небезпеки поразки персоналу електричним струмом, оскільки відносна вологість повітря не перевищує 75%, температура не більш 35°С, відсутні хімічно агресивні

середовища. Живлення електроприладів усередині приміщення здійснюється від двухфазної мережі з заземленою нейтралю напругою 220 В і частотою 50 Гц із використанням автоматів токового захисту. У приміщенні повинна бути застосована схема заземлення. Ураження людини електричним струмом може відбутися у випадку:

1) дотику до відкритих струмоведучих частин;

2) у результаті дотику до струмопровідних неструмоведучих елементів устаткування, що опинилися під напругою в результаті порушення ізоляції або з інших причин.

Заземлення повинно бути зроблено за допомогою гнучкого сплетеного мідного проводу діаметром порядку 1,5 мм2

Для зменшення значень напруг дотику і відповідних їм величин струмів, при нормальному й аварійному режимах роботи устаткування необхідно виконати повторне захисне заземлення нульового проводу.

Відповідно до ГОСТ-12.2.007.0-75 устаткування (крім ЕОМ - II клас) відноситься до I класу, воно має робочу ізоляцію відповідно до вимог ГОСТ 12.1.009-76. Підключення устаткування виконане відповідно до вимог ПБЕ та ПУЕ. Додаткових заходів по електробезпечності не потрібно.

3.6 Пожежна безпека

Робоче приміщення відповідно до ПБЕ та ОНТП 24–86 по вибухово-пожарній безпеці можна віднести до категорії "В".

Можливими причинами виникнення пожежі в приміщенні є:

1) коротке замикання проводки;
2) користування побутовими електрорадіоприладами.
3) не дотримання умов протипожежної безпеки.

У зв'язку з цим відповідно до ПУЕ необхідно передбачити наступні заходи щодо пожежної безпеки: ретельна ізоляція всіх струмоведучих провідників до робочих місць; періодичний огляд і перевірка ізоляції; суворе дотримання норм протипожежної безпеки на робочому місці.

Для гасіння пожеж на робочому місці користувача ПК використовують вуглекислотні та порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються як ручні (ВВК-5);

– Порошкові вогнегасники ВП-2, ВП-5, ВП-10 та ін.

Рисунок 3.1. Засоби пожежогасіння

З метою своєчасного оповіщення, на дільниці необхідно встановити протипожежну сигналізацію. Проходи та запасні виходи повинні бути вільними. Пожежний щит повинен розміщуватись в доступному місці та містити первинні засоби пожежогасіння: вогнегасник, лопату, відро, простирадло, ящик з піском. Відповідальний за пожежну безпеку керівник виробничої дільниці.

ВИСНОВКИ

Під час виконання дипломного проекту були проаналізовані основні алгоритми симетричного шифрування даних, визначені їх переваги та недоліки, методи щодо їх злому, обрані швидкі потокові алгоритми кодування, а також псевдогенератори випадкових чисел для реалізації констант та ключів. За допомогою програмного забезпечення для моделювання криптографічних алгоритмів CrypTool2 виконано моделювання роботи криптографічного алгоритму ChaCha, що є одним з найбільш ефективних та захищених алгоритмів симетричного потокового шифрування в даний час. Це дозволило виконати розробку програмної моделі кодеку ChaCha мовою C++ засобами інтегрованого середовища розробки QT.

Після проведеного аналізу криптографічних та криптоаналітичних методів можна зазначити, що проблема створення надійного алгоритму шифрування буде завжди на високому рівні дослідження та вирішення, адже з розвитком обчислювальних можливостей комп'ютерної техніки зменшуються періоди для встановлення вразливостей та колізій для крипто-алгоритмів.

Вивчені при розробці програмного забезпечення методи шифрування можуть бути корисними не тільки для захисту локальних даних користувача, але і для створення у майбутньому web-сервісу для захисту хмарних даних. Розроблений кодек має візуальний інтерфейс для тестування роботи криптографічного алгоритму ChaCha, додаток підтримується операційними системами сімейства Windows та не вимагає багато ресурсів для своєї роботи.

Подальше вдосконалення роботи програмного забезпечення розробленого кодеку може полягати у додаванні можливості вибору розміру блоків для шифрування даних, застосуванні алгоритмів шифрування для передачі ключа та вектору ініціалізації, збільшенні їх розмірів, надійному зберіганні, можливості використання генераторів випадкових чисел, які будуть залежати від величин, що не можливо передбачити, збільшенні циклів кодування.

Розроблене програмне забезпечення представлене у вигляді моделі для реалізації симетричних потокових алгоритмів шифрування даних, в якій можна впроваджувати нові технології та напрацювання.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глинчук Л.Я. Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук – Луцьк: Вежа-Друк, 2014. – 164 с.
2. Рябко Б. Я., Фионов А. Н. «Криптография в информационном мире» Горячая линия-Телеком, 2018. – 300 с.
3. Горбенко І.Д. Прикладна криптологія. Теорія. Практика. Застосування / І.Д. Горбенко, Ю.І. Горбенко. – Х.:Форт, 2012. – 870 с.
4. Authenticated encryption. [Електронний ресурс]. https://www.cryptopp.com/wiki/Authenticated_Encryption
5. Symmetric And Asymmetric Key Cryptography: All You Need To Know In 3 Points. [Електронний ресурс]. https://www.jigsawacademy.com/blogs/cyber-security/symmetric-and-asymmetric-key-cryptography/
6. Blum-Blum-Shub generator и его применение. [Електронний ресурс]. https://habr.com/ru/post/534698/
7. Secure Hash Algorithms. [Електронний ресурс]. https://brilliant.org/wiki/secure-hashing- algorithms/
8. SHA-512 Hashing Algorithm Overview. [Електронний ресурс]. https://komodoplatform.com/en/blog/sha-512/
9. Ковалюк Т. В. Алгоритмізація та програмування: підручник з грифом МОН України / Т. В. Ковалюк. – Львів : Магнолія-2006, 2013. – 400 с.
10. Богач І. В., Довгалець С. М., Дубовой В. М. Алгоритми розв'язання задач з програмування. Розв'язник. Вінниця: ВНТУ, 2017. – 119 с.
11. Бублик В. В. Об'єктно-орієнтоване програмування. Київ : ІТ-книга, 2015. – 640 с.
12. Шлее, Макс Qt 4.8. Профессиональное программирование на C++ / Макс Шлее. – М.: БХВ-Петербург, 2012. – 894 с.
13. Полубенцева, М. C/C++. Процедурное программирование / М. Полубенцева. – М.: БХВ-Петербург, 2014. – 448 с.
14. Кравець П. Об'єктно-орієнтоване програмування : навч. посібник / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.

ДОДАТОК А. Код основного класу ChaCha мовою C++

```
#include <cstdint>
#include <limits>

template<size_t R>
class ChaCha {
public:
   typedef uint32_t result_type;

   explicit ChaCha(uint64_t seedval, uint64_t stream = 0);
   template<class Sseq> explicit ChaCha(Sseq& seq);

   void seed(uint64_t seedval, uint64_t stream = 0);
   template<class Sseq> void seed(Sseq& seq);

   uint32_t operator()();
   void discard(unsigned long long n);

   template<size_t R_> friend bool operator==(const ChaCha<R_>& lhs, const ChaCha<R_>&
rhs);
   template<size_t R_> friend bool operator!=(const ChaCha<R_>& lhs, const ChaCha<R_>& rhs);

   template<typename CharT, typename Traits>
   friend std::basic_ostream<CharT, Traits>& operator<<(std::basic_ostream<CharT, Traits>& os,
const ChaCha<R>& rng);

   template<typename CharT, typename Traits>
   friend std::basic_istream<CharT, Traits>& operator>>(std::basic_istream<CharT, Traits>& is,
ChaCha<R>& rng);

   static constexpr uint32_t min() { return std::numeric_limits<uint32_t>::min(); }
   static constexpr uint32_t max() { return std::numeric_limits<uint32_t>::max(); }

private:
   void generate_block();
   void chacha_core();

   alignas(16) uint32_t block[16];
   uint32_t keysetup[8];
   uint64_t ctr;
};

template<size_t R>
inline ChaCha<R>::ChaCha(uint64_t seedval, uint64_t stream) {
   seed(seedval, stream);
}

template<size_t R>
template<class Sseq>
inline ChaCha<R>::ChaCha(Sseq& seq) {
   seed(seq);
}
```

```
template<size_t R>
inline void ChaCha<R>::seed(uint64_t seedval, uint64_t stream) {
   ctr = 0;
   keysetup[0] = seedval & 0xffffffffu;
   keysetup[1] = seedval >> 32;
   keysetup[2] = keysetup[3] = 0xdeadbeef;
   keysetup[4] = stream & 0xffffffffu;
   keysetup[5] = stream >> 32;
   keysetup[6] = keysetup[7] = 0xdeadbeef;
}

template<size_t R>
template<class Sseq>
inline void ChaCha<R>::seed(Sseq& seq) {
   ctr = 0;
   seq.generate(keysetup, keysetup + 8);
}

template<size_t R>
inline uint32_t ChaCha<R>::operator()() {
   int idx = ctr % 16;
   if (idx == 0) generate_block();
   ++ctr;

   return block[idx];
}

template<size_t R>
inline void ChaCha<R>::discard(unsigned long long n) {
   int idx = ctr % 16;
   ctr += n;
   if (idx + n >= 16 && ctr % 16 != 0) generate_block();
}

template<size_t R>
inline void ChaCha<R>::generate_block() {
   uint32_t constants[4] = {0x61707865, 0x3320646e, 0x79622d32, 0x6b206574};

   uint32_t input[16];
   for (int i = 0; i < 4; ++i) input[i] = constants[i];
   for (int i = 0; i < 8; ++i) input[4 + i] = keysetup[i];
   input[12] = (ctr / 16) & 0xffffffffu;
   input[13] = (ctr / 16) >> 32;
   input[14] = input[15] = 0xdeadbeef;

   for (int i = 0; i < 16; ++i) block[i] = input[i];
   chacha_core();
   for (int i = 0; i < 16; ++i) block[i] += input[i];
}

#ifdef __SSE2__
#include "emmintrin.h"

#if !defined(__XOP__)
   #if defined(__SSSE3__)
```

```
    #include <tmmintrin.h>
    #define _mm_roti_epi32(r, c) (                          \
      ((c) == 8) ?                                  \
        _mm_shuffle_epi8((r), _mm_set_epi8(14, 13, 12, 15,  \
                                           10,  9,  8, 11,  \
                                            6,  5,  4,  7,  \
                                            2,  1,  0,  3)) \
      : ((c) == 16) ?                               \
        _mm_shuffle_epi8((r), _mm_set_epi8(13, 12, 15, 14,  \
                                            9,  8, 11, 10,  \
                                            5,  4,  7,  6,  \
                                            1,  0,  3,  2)) \
      : ((c) == 24) ?                               \
        _mm_shuffle_epi8((r), _mm_set_epi8(12, 15, 14, 13,  \
                                            8, 11, 10,  9,  \
                                            4,  7,  6,  5,  \
                                            0,  3,  2,  1)) \
      :                                             \
        _mm_xor_si128(_mm_slli_epi32((r), (c)),             \
                      _mm_srli_epi32((r), 32-(c)))          \
    )
  #else
    #define _mm_roti_epi32(r, c) _mm_xor_si128(_mm_slli_epi32((r), (c)), \
                                       _mm_srli_epi32((r), 32-(c)))
  #endif
#else
  #include <xopintrin.h>
#endif

template<size_t R>
inline void ChaCha<R>::chacha_core() {
  // ROTVn rotates the elements in the given vector n places to the left.
  #define CHACHA_ROTV1(x) _mm_shuffle_epi32((__m128i) x, 0x39)
  #define CHACHA_ROTV2(x) _mm_shuffle_epi32((__m128i) x, 0x4e)
  #define CHACHA_ROTV3(x) _mm_shuffle_epi32((__m128i) x, 0x93)

  __m128i a = _mm_load_si128((__m128i*) (block));
  __m128i b = _mm_load_si128((__m128i*) (block + 4));
  __m128i c = _mm_load_si128((__m128i*) (block + 8));
  __m128i d = _mm_load_si128((__m128i*) (block + 12));

  for (int i = 0; i < R; i += 2) {
    a = _mm_add_epi32(a, b);
    d = _mm_xor_si128(d, a);
    d = _mm_roti_epi32(d, 16);
    c = _mm_add_epi32(c, d);
    b = _mm_xor_si128(b, c);
    b = _mm_roti_epi32(b, 12);
    a = _mm_add_epi32(a, b);
    d = _mm_xor_si128(d, a);
    d = _mm_roti_epi32(d, 8);
    c = _mm_add_epi32(c, d);
    b = _mm_xor_si128(b, c);
    b = _mm_roti_epi32(b, 7);
```

```
      b = CHACHA_ROTV1(b);
      c = CHACHA_ROTV2(c);
      d = CHACHA_ROTV3(d);

      a = _mm_add_epi32(a, b);
      d = _mm_xor_si128(d, a);
      d = _mm_roti_epi32(d, 16);
      c = _mm_add_epi32(c, d);
      b = _mm_xor_si128(b, c);
      b = _mm_roti_epi32(b, 12);
      a = _mm_add_epi32(a, b);
      d = _mm_xor_si128(d, a);
      d = _mm_roti_epi32(d, 8);
      c = _mm_add_epi32(c, d);
      b = _mm_xor_si128(b, c);
      b = _mm_roti_epi32(b, 7);

      b = CHACHA_ROTV3(b);
      c = CHACHA_ROTV2(c);
      d = CHACHA_ROTV1(d);
    }

    _mm_store_si128((__m128i*) (block), a);
    _mm_store_si128((__m128i*) (block + 4), b);
    _mm_store_si128((__m128i*) (block + 8), c);
    _mm_store_si128((__m128i*) (block + 12), d);

    #undef CHACHA_ROTV3
    #undef CHACHA_ROTV2
    #undef CHACHA_ROTV1
}
#else
template<size_t R>
inline void ChaCha<R>::chacha_core() {
    #define CHACHA_ROTL32(x, n) (((x) << (n)) | ((x) >> (32 - (n))))

    #define CHACHA_QUARTERROUND(x, a, b, c, d) \
      x[a] = x[a] + x[b]; x[d] ^= x[a]; x[d] = CHACHA_ROTL32(x[d], 16); \
      x[c] = x[c] + x[d]; x[b] ^= x[c]; x[b] = CHACHA_ROTL32(x[b], 12); \
      x[a] = x[a] + x[b]; x[d] ^= x[a]; x[d] = CHACHA_ROTL32(x[d],  8); \
      x[c] = x[c] + x[d]; x[b] ^= x[c]; x[b] = CHACHA_ROTL32(x[b],  7)

    for (int i = 0; i < R; i += 2) {
      CHACHA_QUARTERROUND(block, 0, 4,  8, 12);
      CHACHA_QUARTERROUND(block, 1, 5,  9, 13);
      CHACHA_QUARTERROUND(block, 2, 6, 10, 14);
      CHACHA_QUARTERROUND(block, 3, 7, 11, 15);
      CHACHA_QUARTERROUND(block, 0, 5, 10, 15);
      CHACHA_QUARTERROUND(block, 1, 6, 11, 12);
      CHACHA_QUARTERROUND(block, 2, 7,  8, 13);
      CHACHA_QUARTERROUND(block, 3, 4,  9, 14);
    }

    #undef CHACHA_QUARTERROUND
    #undef CHACHA_ROTL32
```

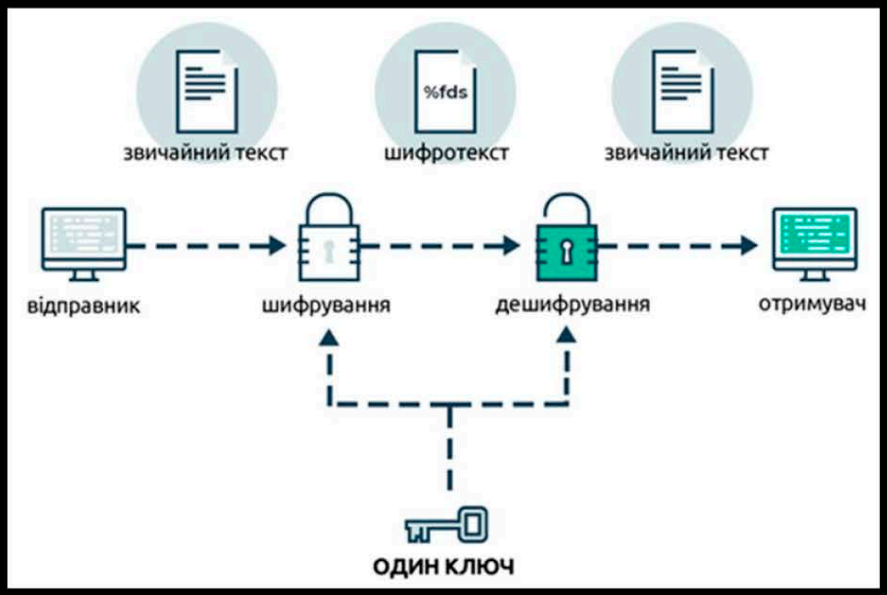
```
}
#endif

template<size_t R>
inline bool operator==(const ChaCha<R>& lhs, const ChaCha<R>& rhs) {
   for (int i = 0; i < 8; ++i) {
      if (lhs.keysetup[i] != rhs.keysetup[i]) return false;
   }

   return lhs.ctr == rhs.ctr;
}

template<size_t R>
inline bool operator!=(const ChaCha<R>& lhs, const ChaCha<R>& rhs) { return !(lhs == rhs); }

template<size_t R, typename CharT, typename Traits>
inline std::basic_ostream<CharT, Traits>& operator<<(std::basic_ostream<CharT, Traits>& os,
const ChaCha<R>& rng) {
   typedef typename std::basic_ostream<CharT, Traits>::ios_base ios_base;

   auto flags = os.flags();
   auto fill = os.fill();

   auto space = os.widen(' ');
   os.flags(ios_base::dec | ios_base::fixed | ios_base::left);
   os.fill(space);

   // Serialize.
   for (int i = 0; i < 8; ++i) os << rng.keysetup[i] << space;
   os << rng.ctr;

   // Sestore old state.
   os.flags(flags);
   os.fill(fill);

   return os;
}

template<size_t R, typename CharT, typename Traits>
inline std::basic_istream<CharT, Traits>& operator>>(std::basic_istream<CharT, Traits>& is,
ChaCha<R>& rng) {
   typedef typename std::basic_istream<CharT, Traits> ::ios_base ios_base;

   auto flags = is.flags();
   is.flags(ios_base::dec);

   for (int i = 0; i < 8; ++i) is >> rng.keysetup[i];
   is >> rng.ctr;

   is.flags(flags);

   return is;
}
```

ДОДАТОК Б. Слайди мультимедійної презентації

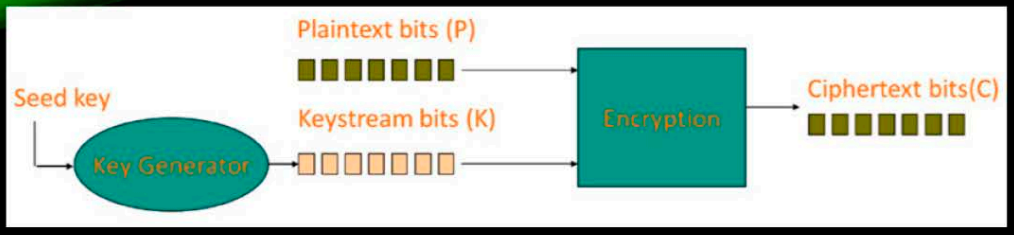
Розробка моделі кодеку на базі алгоритму симетричного шифрування

Тімофєєва Валерія, ВСП "ОТФК ОНТУ"

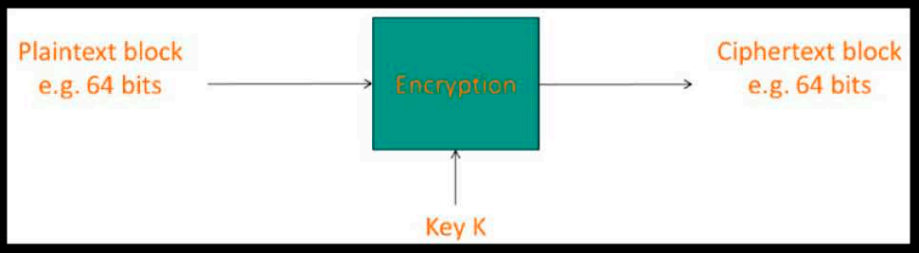
Узагальнена схема симетричного шифрування



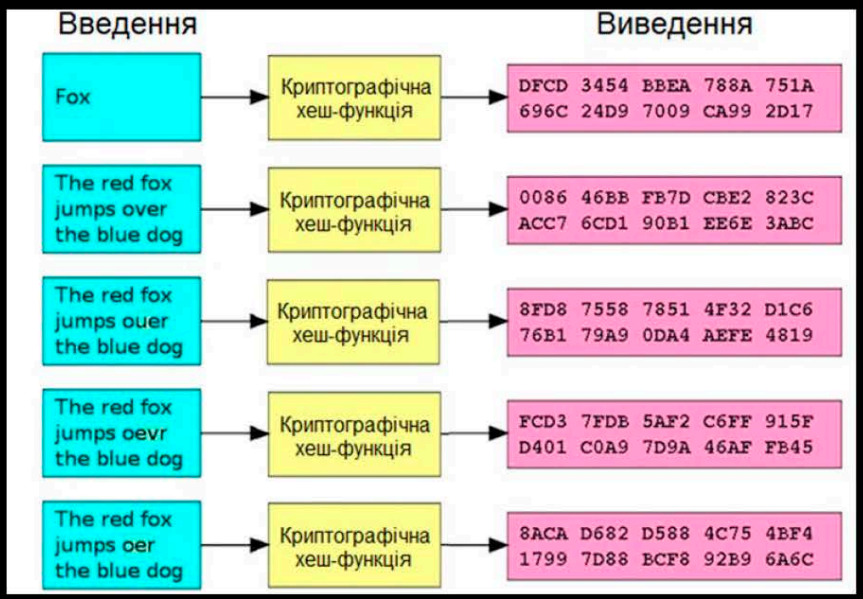
Принципи потокового шифрування



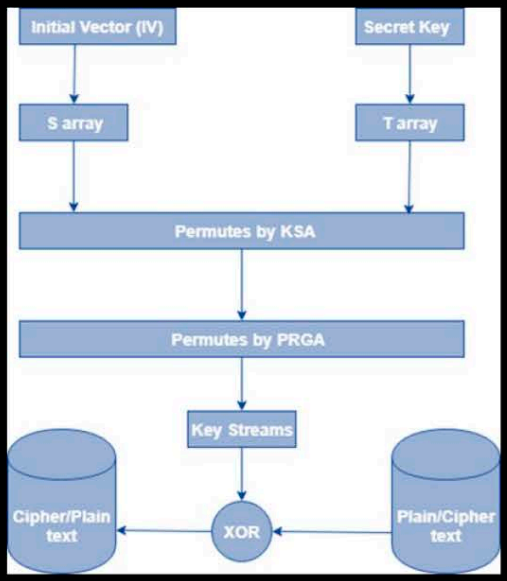
Принципи блочного шифрування



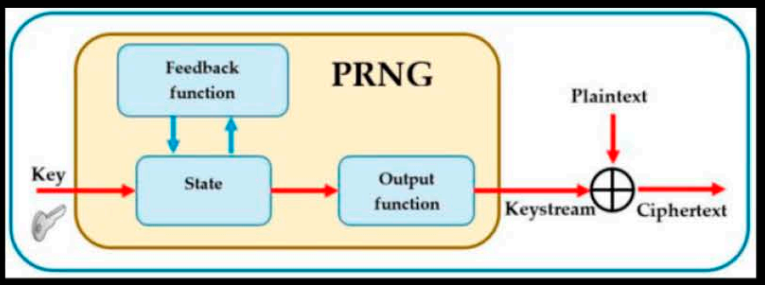
Принцип хешування за допомогою хеш-функції



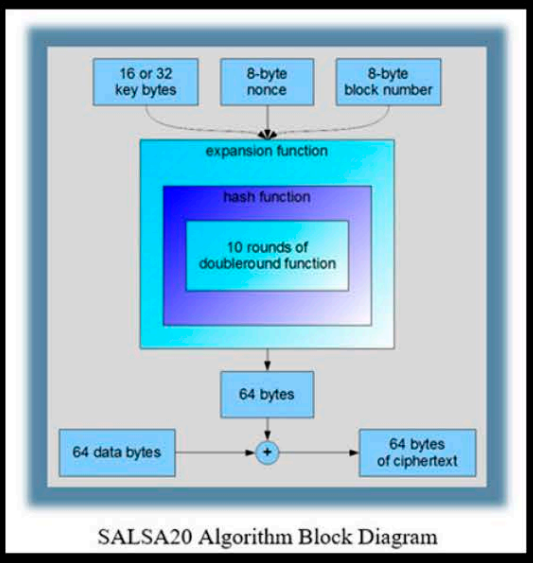
Реалізація шифрування за алгоритмом RC4



Генерування псевдовипадкового ключу

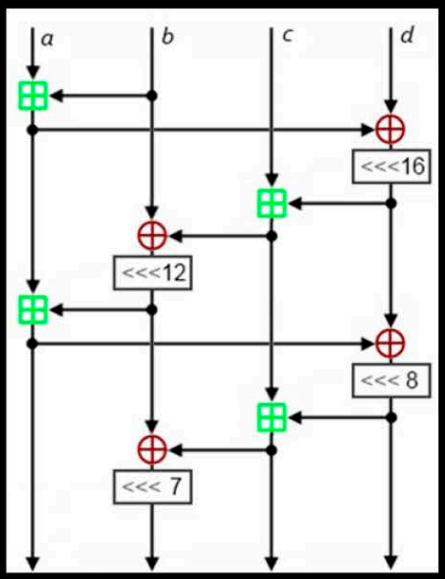


Блок-схеми алгоритму SALSA20, на якому заснований кодек ChaCha



SALSA20 Algorithm Block Diagram

Цикл роботи алгоритму ChaCha функції чверті раунду quarterround(a, b, c, d)



$$a+=b;\quad d\oplus=a;\quad d\lll=16;$$
$$c+=d;\quad b\oplus=c;\quad b\lll=12;$$
$$a+=b;\quad d\oplus=a;\quad d\lll=8;$$
$$c+=d;\quad b\oplus=c;\quad b\lll=7;$$

Спрощена схема вихорю Мерсенна

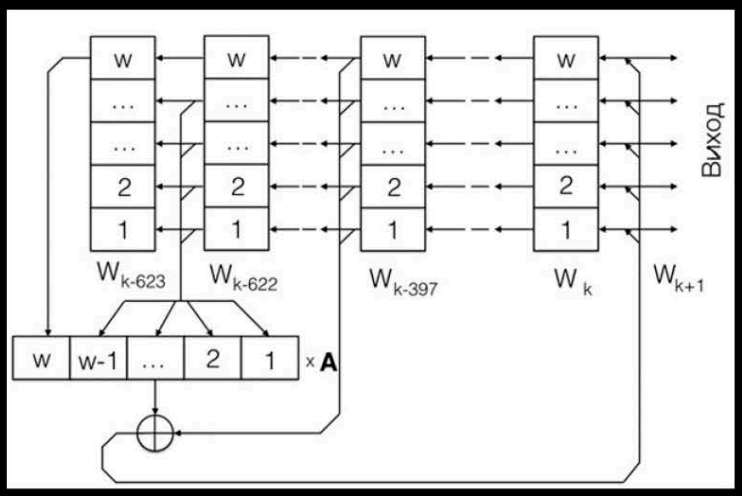
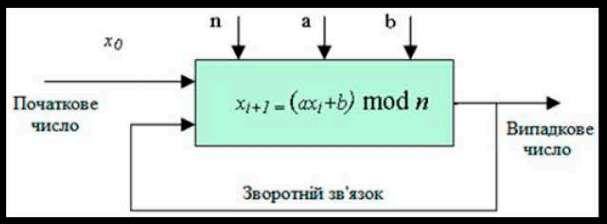


Схема реалізації алгоритму BBS



Структура алгоритму SHA-512

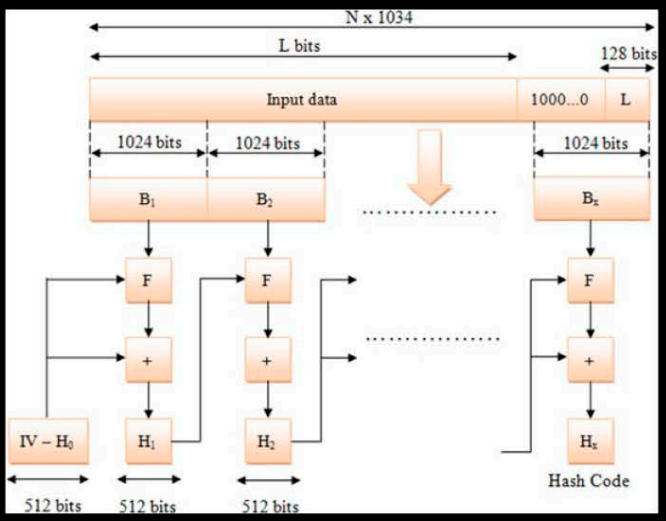
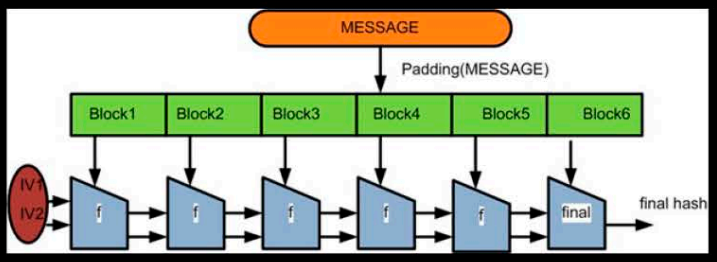
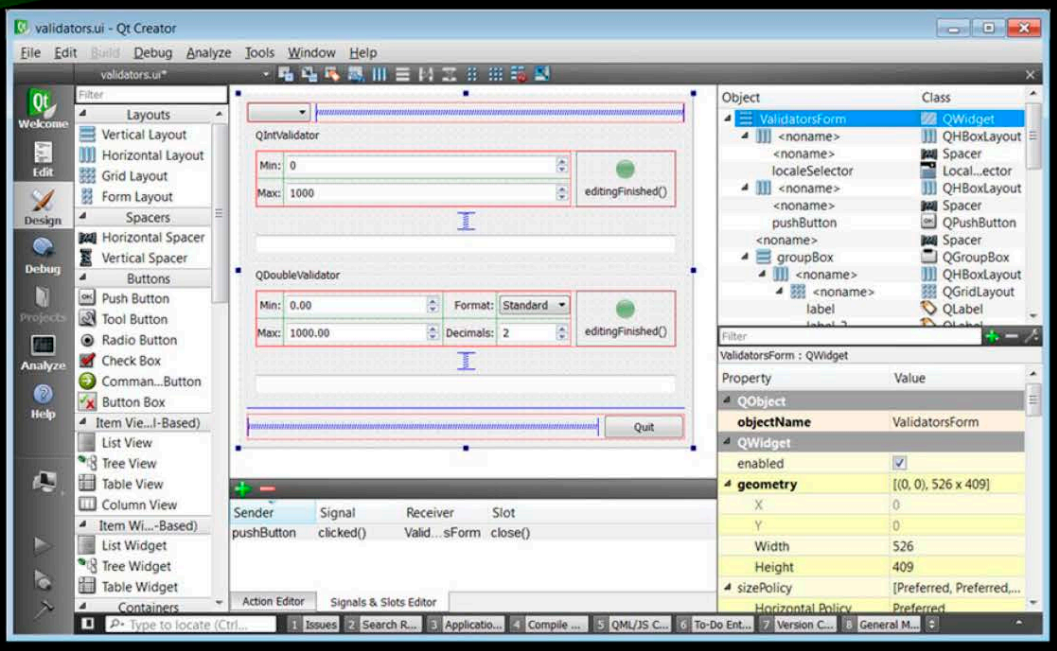


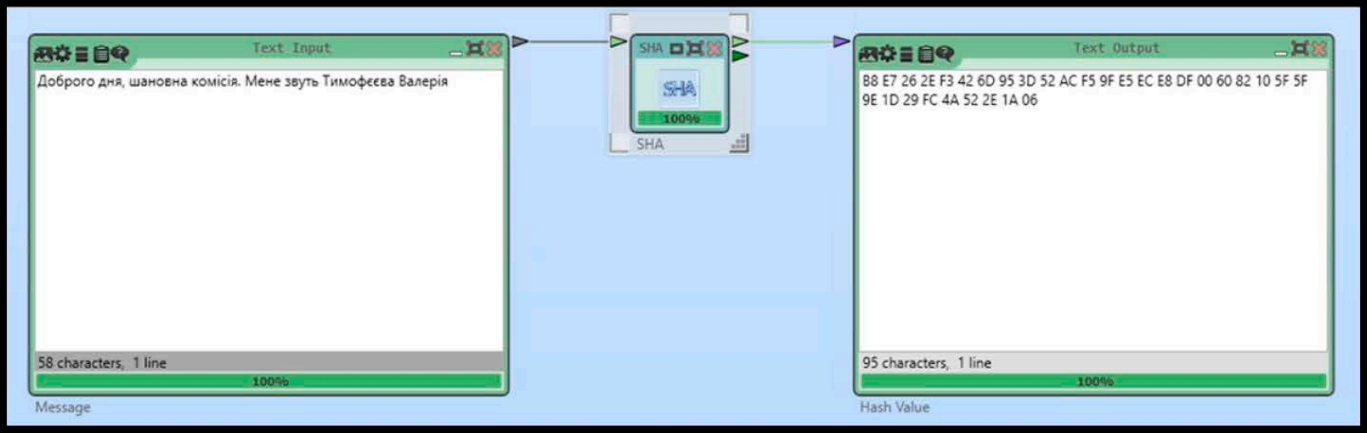
Схема реалізації алгоритму Меркла-Демґарда для хешування



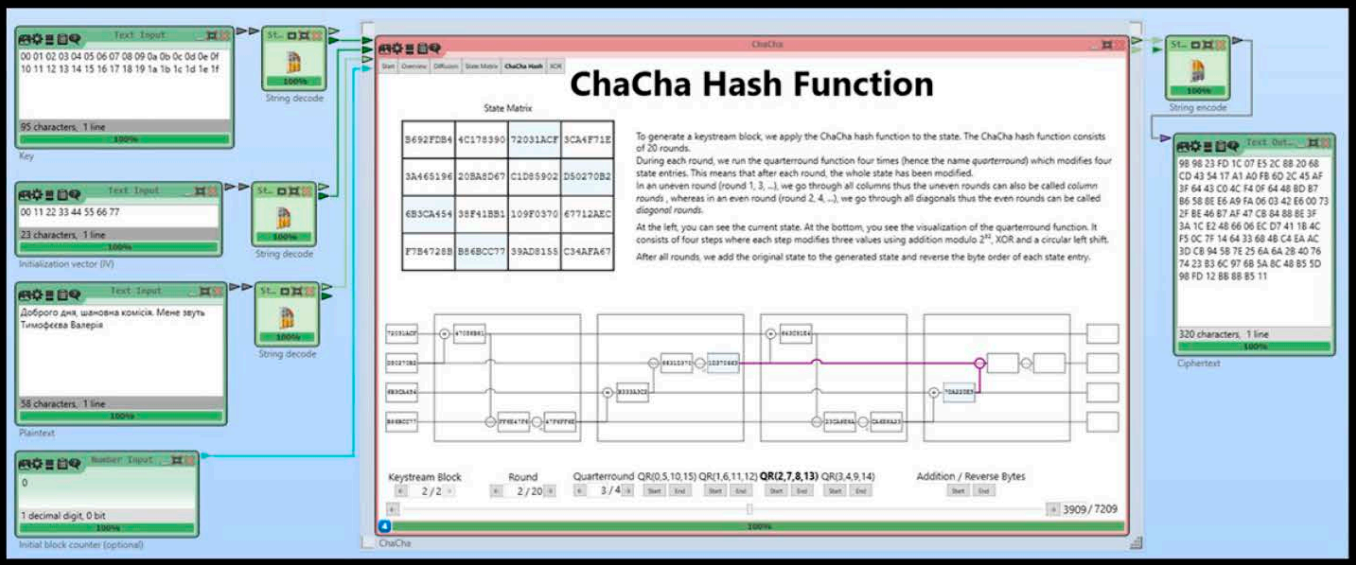
Інтерфейс IDE QT Creator



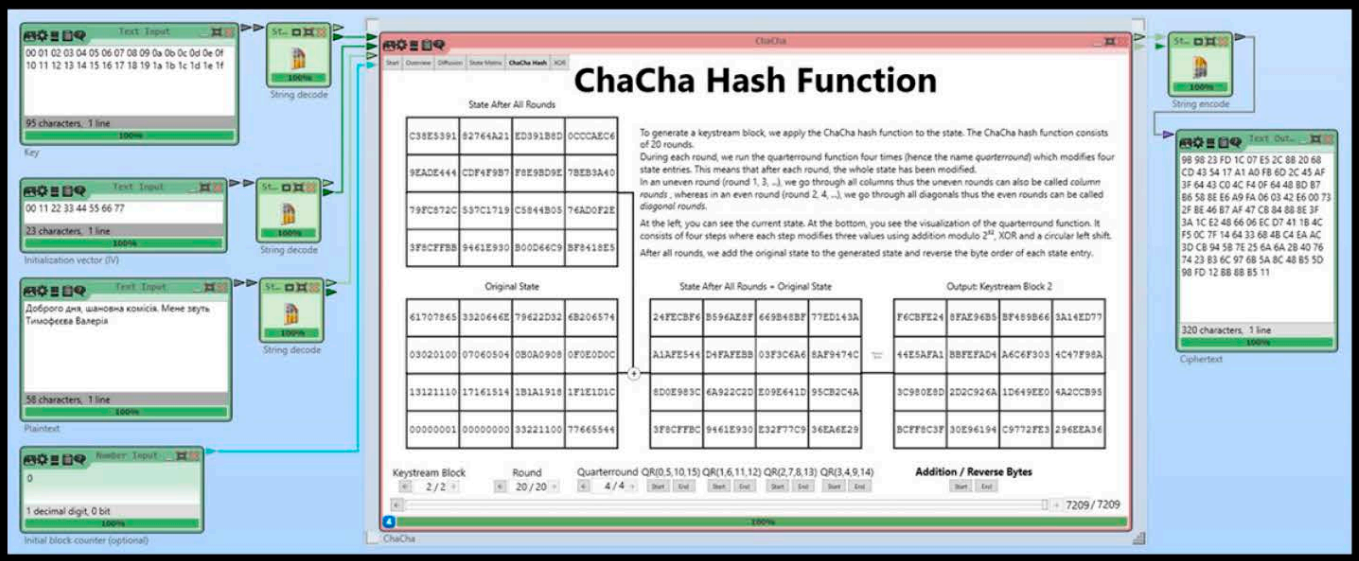
Результат моделювання за алгоритмом SHA-512



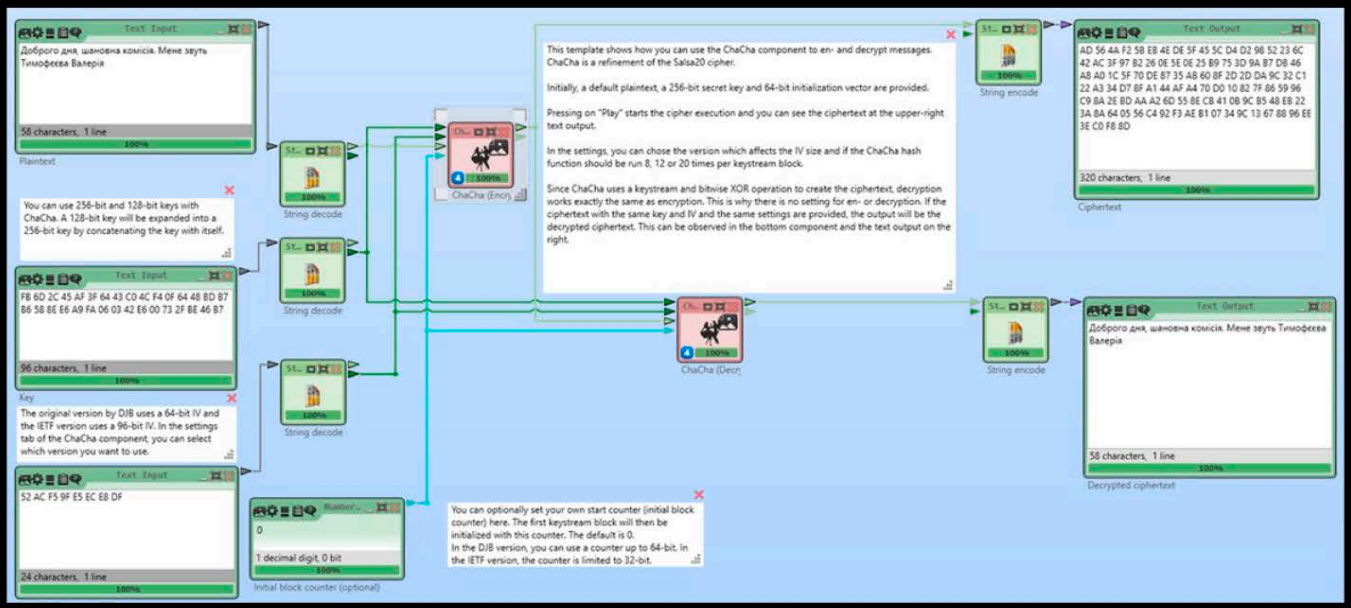
Результат моделювання шифрування за алгоритмом ChaCha на кроці 3909



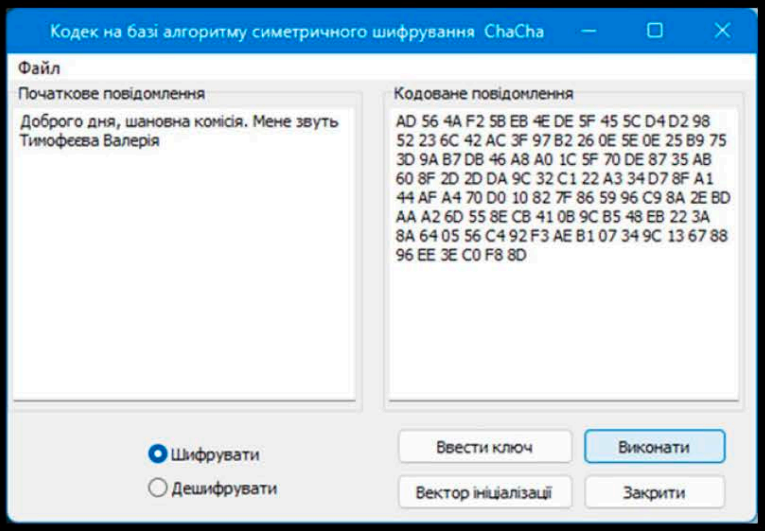
Результат моделювання шифрування за алгоритмом ChaCha на кроці 7209



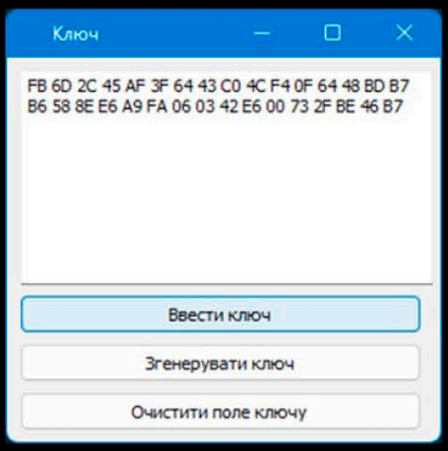
Результат моделювання шифрування та дешифрування за алгоритмом ChaCha

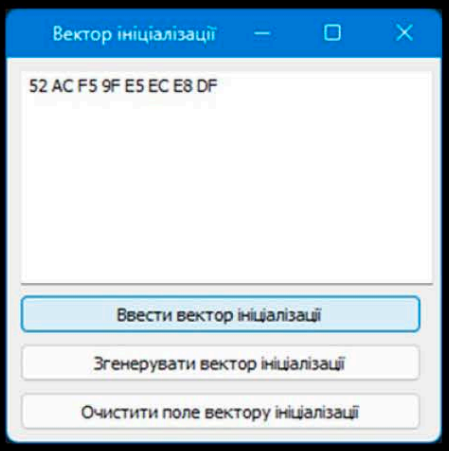


Головне вікно програми-кодеку на базі алгоритму ChaCha

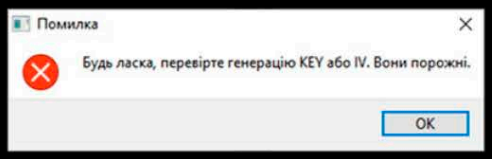


Генерування або введення ключу та вектору ініціалізації

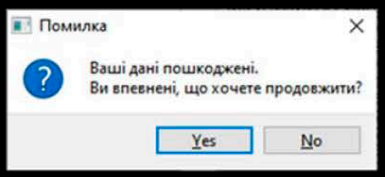




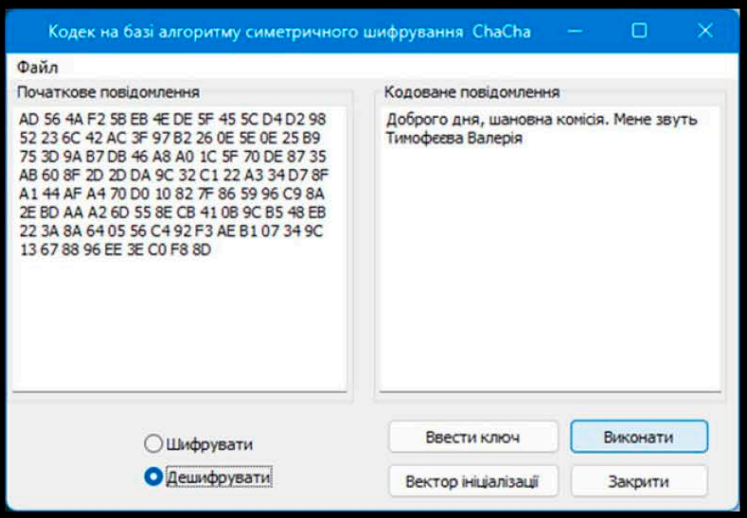
Повідомлення про відсутність ключу / вектору ініціалізації



Повідомлення про порушення цілісності даних



Приклад дешифрування програмою-кодеком повідомлення



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Тімофєєвої Валерії Леонідівни

(прізвище, ім`я та по батькові)

Спеціальність 123 "Комп'ютерна інженерія"

Освітня програма «Комп'ютерна графіка і Web-дизайн»

Керівник дипломного проекту (роботи) Кривченко Юрій Вікторович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) Розробка моделі кодеку на базі алгоритму симетричного шифрування

Обсяг розрахунково-пояснювальної записки 71 сторінок

Обсяг графічної (презентаційної) частини 18 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною та присвячена етапам розробки моделі кодеку на базі алгоритму симетричного шифрування ChaCha з застосуванням рівномірного розподілу.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У технологічному розділі виконано огляд і аналіз властивостей алгоритмів симетричного шифрування, огляд сучасних алгоритмів шифрування, аналіз технічного завдання і постановка задачі проектування, розробка моделі кодеку та його програмна реалізація, розробка програмного коду та створення інтерфейсу.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

Вивчені при розробці програмного забезпечення методи шифрування можуть бути корисними не тільки для захисту локальних даних користувача, але і для створення у майбутньому web-сервісу для захисту хмарних даних. Розроблений кодек має візуальний інтерфейс для тестування роботи криптографічного алгоритму ChaCha.

д) основні недоліки дипломного проекту (роботи)

Серед недоліків роботи варто вказати, що не передбачено додавання можливості вибору розміру блоків для шифрування даних та не розглянуто застосування алгоритмів шифрування для передачі ключа.

Оцінка розрахункової частини відмінно

Оцінка графічної частини відмінно

Загальна оцінка відмінно

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента

"Державний університет інтелектуальних технологій і зв'язку",
доцент кафедри кібербезпеки та технічного захисту інформації,
помічник декана факультету інформаційних технологій та кібербезпеки

Підпис:

«16» червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО НАУКИ І ОСВІТИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ЗВ'ЯЗКУ * ВІДДІЛ КАДРІВ * 43997335 *

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Тімофєєвої Валерії Леонідівни

(прізвище, ім`я та по батькові)

Спеціальність: 123 "Комп'ютерна інженерія"

Освітня програма: «Комп'ютерна графіка і Web-дизайн»

Тема дипломного проекту: Розробка моделі кодеку на базі алгоритму симетричного шифрування

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 71 сторінку. У пояснювальній записці наведено етапи розробки моделі кодеку на базі алгоритму симетричного шифрування ChaCha. Графічна частина складається з 18 слайдів мультимедійної презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.

б) самостійність роботи над проектом: Протягом всього строку дипломного проектування та переддипломної практики здобувачка освіти Тімофєєва В.Л. поступово та послідовно виконувала всі етапи розробки. Всі роботи студентка виконувала самостійно, з оглядом на рекомендації керівника

в) теоретична підготовка випускника (випускниці):

Здобувачка освіти Тімофєєва В.Л. під час роботи над дипломним проектом вивчила достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка дипломниці достатня і вона готова до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання

Під час дипломного проектування здобувачка освіти Тімофєєва В.Л. мала змогу самостійно приймати окремі рішення з реалізації програмної моделі та показала вміння організовано працювати над поставленим завданням, використовуючи сучасні програмні засоби розробки, зокрема інтегроване середовище розробки QT

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові керівника дипломного проекту

Кривченко Юрій Вікторович

Місце роботи і посада керівника дипломного проекту

ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії, голова ЦК КТ та ПІ

Підпис

«8» червня 2023 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Тімофєєва Валерія Леонідівна,
здобувач освіти гр. 4КГ-06, та

Кривченко Юрій Вікторович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи молодшого спеціаліста на тему:

«Розробка моделі кодеку на базі алгоритму симетричного шифрування» (автор роботи – Тімофєєва В.Л., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Тімофєєва В.Л. /

Керівник _____ / Кривченко Ю.В. /

« 12 » червня 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015285888

Дата перевірки:
27.05.2023 15:49:20 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
27.05.2023 15:53:40 EEST

ID користувача:
100011688

Назва документа: Тімофєєва В.Л._4КГ-06

Кількість сторінок: 52 Кількість слів: 9440 Кількість символів: 69314 Розмір файлу: 1.95 MB ID файлу: 1014958390

20.4%
Схожість

Найбільша схожість: 6.53% з Інтернет-джерелом (https://ela.kpi.ua/handle/123456789/43317)

20.4% Джерела з Інтернету	581	Сторінка 54

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	27