НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»

Завідувач кафедри
_____Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" червня 2021 р.

Дипломний проєкт

на здобуття ступеня бакалавра

зі спеціальності **123 «Комп'ютерна інженерія»**

на тему: Алгоритм шифрування даних на базі Salsa20

Виконала:

студентка IV курсу, групи КВ-73
(шифр групи)

Білоха Анна Кирилівна _____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц. каф. СПіСКС, к. т. н., доцент Тарасенко-Клятченко О.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент ст.викл. каф. ПЗКС, к.т.н. Хицко Я.В. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» _____2021 р.

ЗАВДАННЯ
на дипломний проєкт студентки
Білоха Анна Кирилівна
(прізвище, ім'я, по батькові)

1. Тема проєкту «Алгоритм шифрування даних на базі Salsa20»,

керівник проєкту доц. каф. СПіСКС, к. т. н., доцент Тарасенко-Клятченко О.В.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 2021 р. №_____

2. Термін подання студентом проєку_____

3. Вихідні дані до проєкту:

- криптостійка система шифрування даних на базі Salsa20 з перевіркою на цілісність даних.

4. Зміст пояснювальної записки

- аналіз існуючих рішень та обґрунтування теми дипломного проєкту;
- аналіз технологій, які використовуються при розробці криптостійких систем;
- опис розробленої криптографічної системи із перевіркою на цілісність даних.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- алгоритм перевірки цілісності даних (схема алгоритму);
- алгоритм Salsa20 (схема алгоритму);
- формування ключа та вектору ініціалізації (схема алгоритму);
- модулі програми (схема структурна).

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., к.т.н., доцент каф. СПіСКС		

7. Дата видачі завдання "20" жовтня 2021 р._____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	20.02.2021	
2.	Розроблення та узгодження технічного завдання	10.03.2021	
3.	Аналіз існуючих рішень	15.03.2021	
4.	Розробка архітектури системи	05.04.2021	
5.	Розробка графічної частини системи	20.04.2021	
6.	Розробка системи шифрування	23.04.2021	
7.	Тестування	04.05.2021	
8.	Оформлення документації дипломного проєкту	05.05.2021	
9.	Підготовка матеріалів графічної частини проєкту	15.05.2021	

Студентка _____ Білоха А.К.
(підпис) (ініціали, прізвище)

Керівник проєкту _____ Тарасенко-Клятченко О.В
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Дипломний проєкт включає у себе пояснювальну записку (51 ст., 15 рис., 2 додатки).

Об'єктом розробки – створення криптостійкої системи шифрування даних на базі алгоритму Salsa20.

Система шифрування дозволяє: шифрувати та дешифрувати інформацію, здійснювати швидку передачу даних з перевіркою на їх цілісність, генерувати криптостійкий нонс та швидкий псевдовипадковий ключ. У процесі розробки було використано програмне середовище QT для створення інтуїтивно-зрозумілого інтерфейсу.

У ході розробки:

- Проведено аналіз існуючих алгоритмів шифрування;
- Проведено аналіз основних кібератак на конфіденційні дані;
- Дослідження технологій розробки для відповідних систем;
- Сформульовані вимоги для криптостійкої системи;
- Розроблено та протестовано систему для кодування та декодування інформації на базі існуючого алгоритму.

Упровадження цієї системи дозволить збільшити швидкість обробки даних для шифрування із забезпеченням захисту від зловмисників.

Ключові слова: криптостійка система, Salsa20, цілісність даних, нонс, пседовипадковий ключ, QT.

ABSTRACT

Diploma project includes an explanatory note (51 p., 15 fig. 2 appendices).

The object of development is to create a cryptographically strong algorithms for data encryption systems based on the Salsa20 algorithm.

The encryption system allows: to encrypt and decrypt information, to perform fast data transfer with verification of their integrity, generate a a cryptographically strong nonce and a fast pseudorandom key. The QT software environment was used in the development process to create an intuitive interface.

During development:

- The analysis of existing encryption algorithms is carried out;
- The analysis of the main cyber attacks on confidential data is carried out;
- Research of development technologies for the corresponding systems;
- The requirements for a cryptographically strong system are formulated;
- Developed and tested a system for encoding and decoding information based on the existing algorithm..

Keywords: strong cryptography, Salsa20, data integrity, nonce, pseudorandom key, QT.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045470.002 ТЗ	Алгоритм шифрування	4		
			даних на базі Salsa20			
			Технічне завдання			
	А4	ІАЛЦ.045470.003 ТП	Алгоритм шифрування	2		
			даних на базі Salsa20			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045470.004 ПЗ	Алгоритм шифрування	50		
			даних на базі Salsa20			
			Пояснювальна записка			
	А4	ІАЛЦ.045470.005 Д1	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Алгоритм перевірки			
			цілісності даних			
			Схема алгоритму			
	А4	ІАЛЦ.045470.006 Д2	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Алгоритм Salsa20			
			Схема алгоритму			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ.045470.001 ОА		
Розроб.		Білоха А.К.			Алгоритм шифрування даних на базі Salsa20 **Опис альбому**	Літ.	Лист 1 / Листів 2
Перев.		Тарасенко-Клятченко О.В.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-73	
Н. контр.		Клятченко Я.М.					
Затв.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045470.007 Д3	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Формування ключа та			
			вектору ініціалізації			
			Схема алгоритму			
	А4	ІАЛЦ.045470.008 Д4	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Модулі програми			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045470.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Алгоритм шифрування даних на базі Salsa20».

Галузь застосування: комп'ютерній техніці в системах приховування конфіденційної і комерційної інформації.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення криптостійкої системи шифрування даних на базі алгоритму Salsa20.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь-якою операційною ситемою: Windows, Linux;
- простий та інтуїтивно-зрозумілий інтерфейс;
- криптостійка та швидка система шифрування;
- можливість шифрувати да дешифрувати дані;

- перевірка на цілісність даних.

5.2. Вимоги до апаратного забезпечення

- Оперативна пам’ять: 2 Гб.

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows, Linux;
- програмне забезпечення qt або наявні бібліотеки у папці з файлом .exe – Qt5Core.dll, Qt5Gui.dll, Qt5Widgets.dll, libgcc_s_seh-1.dll, libstdc++-6.dll, qt.conf, \plugins\platforms\qminimal.dll, \plugins\platforms\qwindows.dll.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	20.02.2021
2.	Розроблення та узгодження технічного завдання	10.03.2021
3.	Аналіз існуючих рішень	15.03.2021
4.	Розробка архітектури системи	05.04.2021
5.	Розробка графічної частини системи	20.04.2021
6.	Розробка системи шифрування	23.04.2021
7.	Тестування	04.05.2021
8.	Оформлення документації дипломного проєкту	05.05.2021
9.	Підготовка матеріалів графічної частини проєкту	15.05.2021

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045470.004 ПЗ	Алгоритм шифрування	51		
			даних на базі Salsa20			
			Пояснювальна записка			
	А4	ІАЛЦ.045470.005 Д1	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Алгоритм перевірки			
			цілісності даних			
			Схема алгоритму			
	А4	ІАЛЦ.045470.006 Д2	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Алгоритм Salsa20			
			Схема алгоритму			
	А4	ІАЛЦ.045470.007 Д3	Алгоритм шифрування	1		
			даних на базі Salsa20			
			Формування ключа та			
			вектору ініціалізації			
			Схема алгоритму			
	А4	ІАЛЦ.045470.008 Д4	Алгоритм шифрування	1		
			даних на базі Salsa20			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045470.003 ТП		
Розробив		Білоха А.К.			Алгоритм шифрування даних на базі Salsa20 **Відомість технічного проєкту**	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Тарасенко-Клятченко О.В.					
Н. контроль		Клятченко Я.М.				**КПІ ім. Ігоря Сікорського, ФПМ, КВ-73**	
Зав. каф.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Модулі програми			
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045470.003 ТП	Арк. 2

Пояснювальна записка
до дипломного проєкту

на тему: Алгоритм шифрування даних на базі Salsa20

Київ – 2021 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Відритий текст – вихідний текст, що зазвичай записаний однією з природних мов, або отриманий в результаті дешифрування шифротексту.

Ентропія – рівень невизначеності, який властивий результатам змінної системи.

Ключ шифрування – набір інформації, за допомогою якої можна зашифрувати або дешифрувати криптографічні дані.

Криптоаналіз – наука про методи перетворення зашифрованої інформації у вихідне значення навіть без доступу до ключа.

Криптографія – наука, яка побудована на створенні та аналізі протоколів кодування даних за допомогою яких, треті особи не мають змоги читати приватну інформацію.

Фреймворк – платформа з програмними рішеннями для розробки систем.

Хеш-функція – перетворення вхідний даних довільного розміру в дані з фіксованим розміром.

Частотний аналіз – застосування статистичних даних появи букв або відповідних груп у певній мові та перенесення їх на картину зашифрованого тексту.

Шифротекст – дані, які були отримані у наслідок шифрування відкритого тексту.

Шифрування – перетворення тексту, який зрозумілий людині у шифротекст.

AES – Advanced Encryption Standard.

CBC – Cipher Block Chaining.

CFB – Cipher Feedback.

CSPRNG – Cryptographically Secure Pseudo-Random Number Generator.

DES (Data Encryption Standard) – стандарт, симетричного алгоритму шифрування, який застосовує мережу Файстеля для інформації блоками 64 біта.

ECB – Electronic Code Book.

FIPS - Federal Information Processing Standards.

IDE (Integrated Drive Electronics) – додаток, для розробки програмного забезпечення.

IPSec – IP Security.

MAC-підпис – коротка інформація для підтвердження відправника і що повідомлення надіслане ним не змінено.

NIST – National Institute of Standards and Technology.

Nonce/IV (Number used once/Initialization vector) – випадкове, довільне число, яке використовується у криптографії лише раз, аби однаковий текст в алгоритмах шифрування закодувати по-різному.

OFB – Output Feedback.

RDRand – команда процесора, яка застосовується для генерації випадкових чисел.

SHA – Secure Hash Algorithm.

SSH – Secure SHell.

SSL (Secure Sockets Layer) – протокол криптографії для безпечного встановлення зв’язку між сервером та клієнтом.

TCP/IP (Transmission Control Protocol/Internet Protocol) – протоколи Інтернету, які складаються з 4 рівнів для врегулювання передачі даних між мережами.

UDP (User Datagram Protocol) – протокол транспортного рівня без підтвердження доставки.

XOR (виняткова диз’юнкція) – додавання двох елементів за модулем два.

mt19937 – Вихор Мерсенна.

ВСТУП

З появою розвинених цивілізацій, об’єднання у групи та становлення ворожнечі серед людей за владу, з’явилася потреба у передачі таємної інформації для збереження секретів між союзниками. Найбільшого прогресу дана сфера досягає сьогодні з появою Інтернету, коли необхідно на великі відстані передавати паролі, особисту інформацію та інше.

Спочатку усвідомлення прийшло до жителів Стародавнього Єгипту, які у 4 тисячолітті до нашої ери почали використовувати ієрогліфи для своєї писемності [1]. Перший алгоритм з’явився у Римі завдяки Юлію Цезарю, що застосовував моноалфавітний шифр зсуваючи літери алфавіту на певну кількість. Проте, такі алгоритми легко розшифрувати застосовуючи частотний аналіз. Саме розуміння криптоаналізу, алгоритмів криптографії є важливою сферою для забезпечення конфіденційності переданої інформації. Адже з’ясування того, як влаштований даний розділ, як працюють та як зламувати алгоритми, лежать в основі створення криптостійких систем.

Найвідоміший приклад криптоаналізу, який врятував мільйони життів та перевернув хід історії – злом машини «Енігма», яка базується на трьох роторах. Тюрінг зміг побудувати машину «Бомба» завдяки роботам польських математиків, та головній вразливості «Енігма» – нездатність букви ставати самою собою у зашифрованому варіанті [2].

Сучасні алгоритми використовують ключі шифрування, які застосовуються у симетричній та асиметричній криптографії для зручності користувачів у захисті даних без знань розділів складної математики.

Основа криптографії: встановити безпечний ключ, який не можна передбачити та безпечне спілкування при наявності спільного ключа. Вільна комунікація має забезпечуватися при наявності прослуховування та підробки даних. Ключ має використовувати генератори випадкових чисел, тобто бути криптостійким, а саме застосовувати рівномірний розподіл на певному інтервалі. Проблема більшості псевдогенераторів, що вони застосовуються на

комп’ютерах, які є детермінованими та використовують конкретно задану константу, тобто не мають джерела достатньої ентропії – кожне наступне число, маючи певний ряд чисел можна передбачити.

У даному бакалаврському проєкті акцентовано увагу на тому, як створити абсолютно стійкі алгоритми шифрування та на реалізації симетричного потокового алгоритму шифрування на основі Salsa20 з застосуванням рівномірного розподілу. Створення відповідного методу кодування може захистити приватну інформацію від сторонніх осіб.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальний опис проблеми

Нині не існує систем, які не можна зламати, питання лише в ресурсах, які необхідно використати для отримання даних. Розробники вдаються до хитрощів аби захистити інформацію. Найбільш криптозахищені системи застосовуються в урядах для збереження секретності від якої залежить національна безпека, що унеможливлює розголошення алгоритмів для широкої громадськості.

Основні проблеми у створенні алгоритмів, які використовують зловмисники при атаках – генерація випадкових чисел, ненадійно захищена передача ключів, фізичні властивості проєктування обчислювальних систем, критерії підтвердження актуальності та повноти даних.

В історії були приклади використання некриптостійких алгоритмів. Браузер Netscape Navigator, який набув популярності в 90-х роках, використовував дві версії для внутрішнього користування з підтримкою 128-бітного шифрування, та для міжнародної спільноти із застосуванням 40-бітного. Але в 1995 році Ян Голдберг та Девід Вагнер змогли прослуховувати захищений SSL через передбачувану техніку визначення псевдовипадкового числа.

Криптографічні системи можна зламати, якщо правильно застосовувати криптоаналіз та знання математики. Наприклад, через парадокс днів народжень у математиці – у групі, яка складається не менше ніж 23 особи, ймовірність збігу дня народження складає більше 50 відсотків. Хакери використовують даний парадокс та визначення хеш-функції, яка складається з n-бітного значення та шукають колізії у випадках зменшення даних при передачах.

Основні стратегії кібератак [3] :

- вичерпний пошук (грубої сили) – перевірка всіх можливих ключів;
- частотний аналіз – знання розподілу послідовностей символів, які зберігаються у відкритому та шифротексті. За допомогою цього методу була зламана «Енігма», що застосовувала поліалфавітний шифр;
- днів народження;
- атака «людина посередині» - зловмисники перехоплюють трафік, зазвичай після запиту на надсилання відкритого ключа, а згодом можуть модифікувати чи підслуховування сторони;
- атака сторонніми каналами – знання фізичних слабких місць реалізацій криптосистем;
- диференціальний аналіз несправностей – вивчення помилок, які виникли у криптосистемі під час атак;
- та інші.

Криптографія вирішує проблеми обмеження доступу до даних, а не проблеми з безпекою.

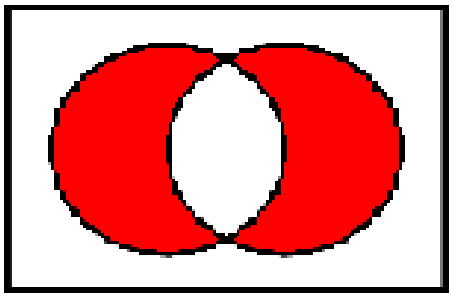
Якщо необхідно захистити дані, то треба використовувати рекомендації затверджені NIST або FIPS. Алгоритми, що пройшли сертифікацію, постійно тестуються, проходять аналізи, застосовуючи математику та статистику, та покращуються.

1.2. Аналіз основних алгоритмів

Майже у всіх алгоритмах застосовується додавання за модулем два XOR ($\oplus$).

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

А



Б

Рисунок 1.1 – Додавання за модулем два: а – у булевій алгебрі;
б – використовуючи діаграму Венна

AES – алгоритм блочного шифрування з відкритим ключем, що використовує 128-бітні дані та різної довжини ключ [4]. Масив байтів записуються у таблицю, де виконуються стандартні математичні операції над окремими елементами масиву (рис. 1.2). Для кожної ітерації даного методу вибирається 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 XOR операція з відповідним байтом ключа ітерації.

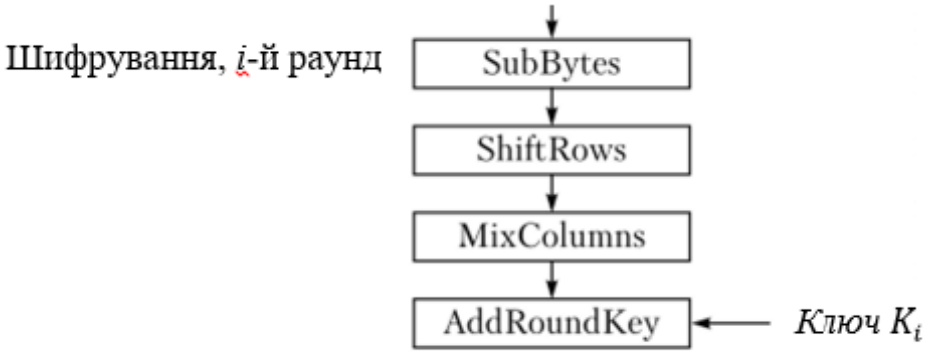


Рисунок 1.2 – Блочна діаграма ітерації шифрування алгоритму AES

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

Недоліки:

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

RC4 – потоковий шифр, який генерує псевдовипадковий потік бітів для ключа та був одним з найбільш використовуваних завдяки простоті реалізації та швидкості роботи, кодуючи великі потоки даних. Він використовує змінні розміри ключів від 40 до 2048 біт.

Вхідні дані – масив байт, ключ – масив бітів. Алгоритм використовує виняткову диз'юнкцію, якщо довжина ключа співмірна з довжиною вхідних даних (рис. 1.3).

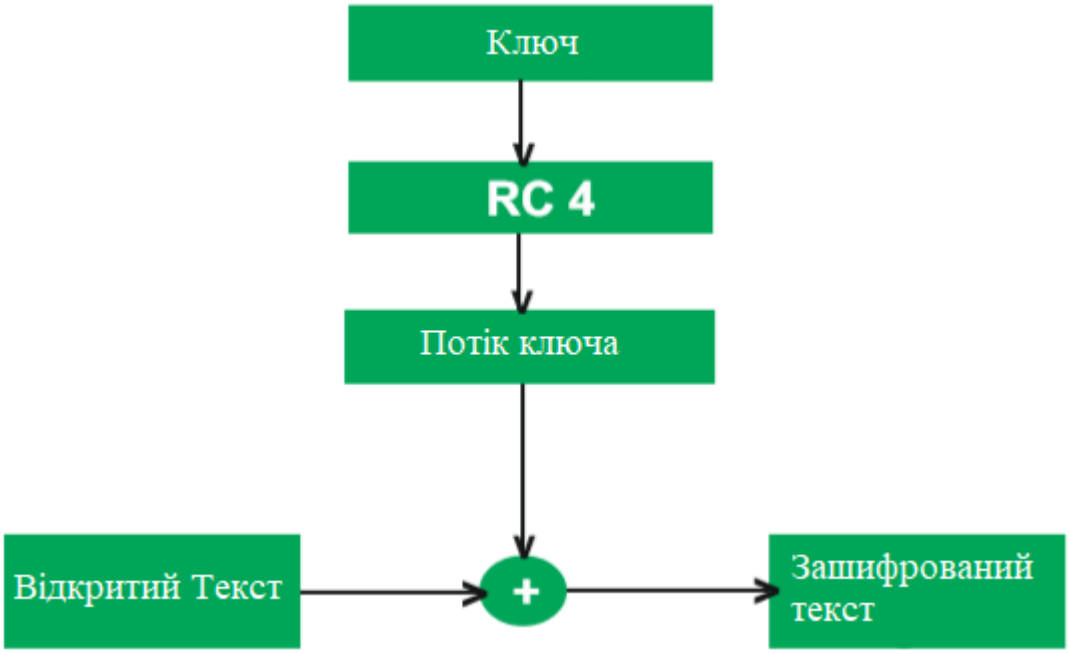


Рисунок 1.3 – Блочна діаграма шифрування алгоритму RC4

Передача таких потоків не виправдана, тому при формуванні кінцевого набору на вхід подається згенерований ключ та застосовуються псевдовипадкові числа на його основі, тобто розширюємо дані. Для їх генерацій застосовується внутрішній стан (рис. 1.4):

1. Перестановка 256 байтів (S-блоки підстановок, позначення «S»).
2. 8-бітові індекси.

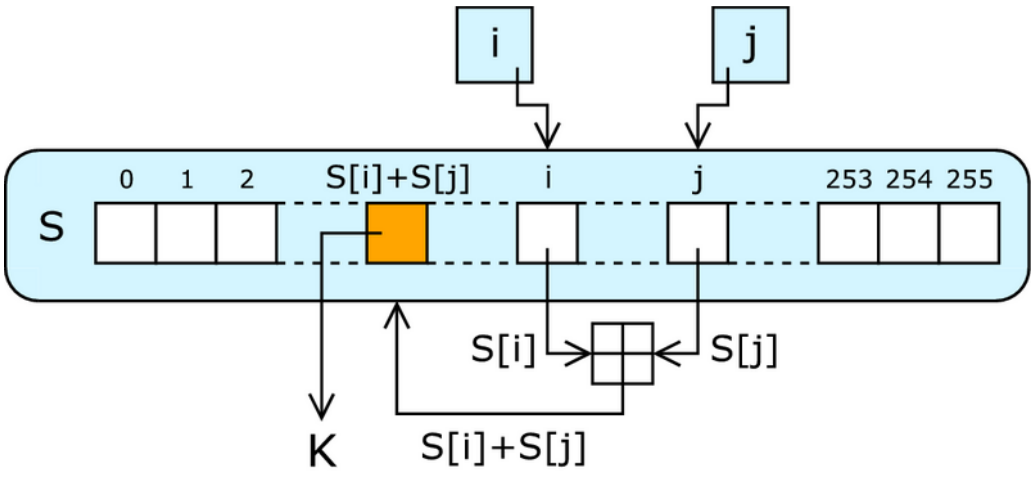


Рисунок 1.4 – Генерація потоку ключів RC4

Недоліком є те, що він не проводить автентифікацію та при застосуванні слабких MAC-підписів зламується за допомогою бітової атаки. Також не можна застосовувати зв'язок між ключами, які згенеровані, оскільки це не відповідає принципам Шеннона. Зараз використовується модифікована версія алгоритму: «RC4-drop[n]».

RSA – перший асиметричний блочний алгоритм, який успішно використано за принципом відкритого ключа для цифрових підписів (рис. 1.5). В його основу покладена факторизація двох великих чисел – розкладення на добуток простих множників. Для вирішення проблеми знаходження відповідних значень застосовуються квантові обчислення, еліптичні криві і алгебраїчна теорія чисел. Якщо у числа існує простий дільник, який відмінний від нього самого, то він не буде перевищувати кореня з числа. Тобто, потрібно перебрати всі числа з проміжку $[2; \sqrt{x}]$ та розділити кожне з них по черзі на x. Якщо у числа немає дільників менших за нього, або які дорівнюють кореню з відповідного, то воно просте [5].

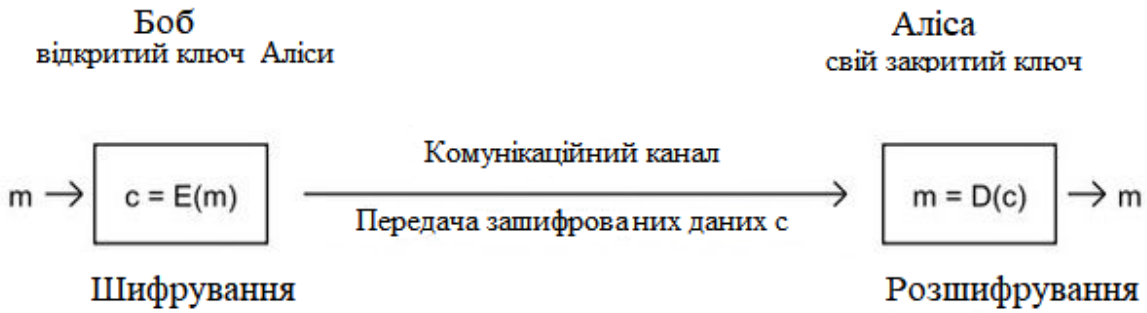


Рисунок 1.5 – Шифрування та розшифрування алгоритмом RSA

RSA використовує односторонню функцію для якої немає ефективних алгоритмів знаходження оборотної.

$$f_{e,n}(x) = x^e \bmod n\,, \qquad (1)$$

де n – цілі числа від 0 до $n-1$.

Застосовуючи дану функцію для алгоритму, на вхід подається відкритий ключ (e, n) та відкритий текст m, а на виході отримуємо шифротекст.

$$c = E(m) = m^e \ mod \ n \ , \tag{2}$$

де n – цілі числа від 0 до $n - 1$;

c – зашифроване повідомлення.

Сьогодні цей метод використовується разом з сеансовими ключами. Інформація, що генерується на основі відкритого та закритого ключа іншої сторони застосовується для захисту каналів зв'язку.

Перевагами методу є те, що його важко зламати, оскільки використовується розкладання на множники простих чисел, тому для дешифрування цього методу необхідно буде використовувати функцію Ейлера. Також, є ефективнішим за будь-який симетричний алгоритм, використовує відкриті ключі, що вирішує основну проблему у безпечній передачі ключів та їх зберігання.

Недоліки: повільний при шифруванні великих даних через велику кількість обчислень, складний в реалізації через надто великий ключ.

Враховуючи дані недоліки розроблених алгоритмів, було створено проєкт для пошуку найкращих потових алгоритмів шифрування – «eSTREAM». До нього увійшли:

- *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;

- *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.

Порівняння швидкостей використання найбільш поширеного RC4 та алгоритмів з проєкту «eSTREAM» на процесорі з базовою тактовою частотою 2.2 GHz та архітектурою x86 (рис. 1.6).

AMD Opteron, 2.2 GHz (Linux)

	PRG	Швидкість (Мбіт/с)
	RC4	126
eStream	Salsa20/12	643
eStream	Sosemanuk	727

Рисунок 1.6 – Порівняння швидкості алгоритмів

Можна побачити, що RC4 повільніший, оскільки не використовує переваг процесорів, а лише виконує операції над байтами.

1.3. Постановка задачі

Задачею даного бакалаврського дипломного проєкту є аналіз актуальності проблеми створення криптостійкої системи, яка б забезпечувала можливість захисту конфіденційної інформації від третіх осіб.

Дану систему вирішено розробляти на основі Salsa20 – найкращому на сьогодні алгоритмі у захисті даних у випадках зберігання на персональному комп’ютері. Використовувалася також платформа QT, для візуально кращого та зрозумілого інтерфейсу для тестування програми, що покращить доступність оцінки розробки.

1.4. Висновки

У даному розділі досліджено та проаналізовано, чи є потреби у створенні нових криптографічних системи для захисту конфіденційної інформації. При аналізі існуючих рішень визначено, що є необхідність вдосконалення наявних

алгоритмів саме через повільну роботу найбільш поширених, пошук нових вразливостей та колізій у системах кодування, розвиток ресурсів обчислювальної здатності зловмисників, які постійно намагаються отримати доступ до перегляду та зміни особистих даних інших користувачів, їх файлів, електронних листів тощо. Хакери докладають зусиль аби дізнатися дані, підмінити їх або пошкодити, змінити автентифікацію.

Розроблений алгоритм буде відрізнятися від базового тим, що використовується швидка генерація ключа, криптостійка псевдовипадкова генерація початкового значення для різного шифрування однакових даних. А також, є перевірка на цілісність інформації за допомогою створення хешу, який використовує не оборотну функцію, що унеможливлює декодування.

РОЗДІЛ 2. ОСНОВНІ ПОНЯТТЯ У КРИПТОГРАФІЇ

2.1. Аналіз криптостійкої системи

Криптографічна стійкість визначається часом та ресурсами, які буде затрачено на те, аби відновити вихідний або відкритий текст. На сьогодні, якщо використовується ключ 256 біт, то це вважатиметься добре захищеною системою, проте, у міру того, що потужність комп'ютерів зростає, з'являється необхідність покращувати алгоритми й надалі.

Ідеальна стійкість була визначена у роботі Клода Шеннона «Теорія зв'язку в секретних системах» у 1946 році. Основними принципами до систем шифрування є вимоги до його ключа [6]:

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

Ці вимоги сформовані навіть при умові, що у зловмисників є необмежені ресурси в обчислювальних системах та криптоаналіз проводиться у відповідності до теорії ймовірностей.

Криптографічні системи можуть бути вразливими до криптографічного аналізу у залежності від їх реалізацій, протоколів, алгоритмів які використовуються.

Найбільш незламним – є шифрування схемою одноразових блокнотів, або шифр Вернама, при правильному використані. На сьогодні, це єдина

система кодування з теоретично доведеною абсолютною криптографічною стійкістю тому, що ця техніка підлягає під основні вимоги, які описав Шеннон. Бінарна версія шифру Моборона і Вернама (3).

$$k = m = c = \{0,1\}^2 , \quad (3)$$

де m – початковий (відкритий) текст;

c – шифротекст;

k – ключ.

Розглянемо приклад: дано початковий текст «HELLO», який не відомий третім особам та які не мають ключа. Для розшифрування, зловмисники починають підбирати порядок даних, які використовує ключ. Якщо ключ був «TQURI», то повідомлення що передавалося, було б: «LATER». Кількість варіантів можливості початкового повідомлення – 26^3:

де 26 – кількість букв, якщо текст складається лише з англійського алфавіту;

3 – кількість комірок, які містять дані букви.

Цей 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 обчислювальних машин полягає у створенні випадкових чисел та передачі ключа такої ж довжини, як вихідний текст, через те що це 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 (4).

$$(m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2 , \quad (4)$$

де m_1, m_2 – відкритий текст;

k – сформований ключ.

У проєкті «Венона» американські криптографи мали змогу розшифрувати тексти тому, що кодова система дала збої і почала використовувати комбінацію однакових цифр декілька разів.

Ще одним правилом розробки криптостійких систем є принцип Керкгоффза, який був викладений у його роботі «Військова криптографія» та опублікований у 1883 році. Ця праця описує шість основних підходів для проєктування військових шифрів [7]:

- 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 телеграфному листуванні.

Тобто, архітектура алгоритму шифрування немає жодним чином впливати на криптографічну стійкість. Це досягається відкритим кодом алгоритмів, для того аби криптоаналітики могли дослідити вразливості методу та виправити відповідні помилки. Адже, з плином часу, будь-які системи підлягають злому з сторони хакерів. Як було наведено раніше, машина «Енігма» була зламана саме через те, що не відповідала принципам Керкгоффза. Цей приклад використовується для наглядного показу механізмів, що приховування даних не забезпечує повний захист.

Повністю протилежним принципом є безпека через неясність, що передбачає приховування важливої інформації. У сфері криптографії це

відбувається відповідними методами: шифруванням даних у бінарному вигляді, зміною порту з 22 на інший, приховування версій ваших програмних забезпечень. Ефективність використання можлива лише у випадках поєднання неясності з іншими механізмами захисту. Військові криптосистеми Шеннона використовують рівень безпеки за допомогою даного методу.

У випадках застосування неясності краще приховувати алгоритми, оскільки для створення нового ключа необхідно менше ресурсів, ніж для нової системи. Проте, зловмисники частіше зламують паролі, бо саме вони можуть дати підказку структури деякої системи, закономірності між механізмами створення випадкових чисел, які можуть застосовуватися в алгоритмах.

2.2. Автентифіковане шифрування

Система повинна забезпечувати не тільки конфіденційність, а й цілісність даних, тому що навіть криптостійкі алгоритми не можуть захистити від пошкодження інформації сторонніми лицями. Автентичність даних (цілісність) означає те, що отримувач при наясвності ключа може перевірити справжність даних.

У 2001 році Гуґо Кравчик навів базові методи автентифікованого шифрування [8]:

- SSL – використовує математичне шифрування, порт 443, тобто створює безпечний канал передачі між двома системами (зазвичай клієнт та сервер), що дає змогу застосовуватися для відправок особистих даних з банків, інформації про кредитні картки, для бізнес цілей, доступу до віддалених програм та іншого;
- SSH – використовує порт 22, що дає змогу підключитися до іншого комп’ютера. Призначений для виконання команд через

віддалений доступ, в той час як SSL 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 TCP, так і для UDP.

У залежності від ситуацій, необхідно використовувати різний протокол. Якщо кількість програм у системі:

- мала, то застосовують автентифікацію за допомогою SSL або SSH;
- велика – IPSec із захистом пакетів та приховуванням пункту призначення.

2.3. Хешування

Хешування – це математична функція, яка на вхід приймає інформацію та перетворює її у стиснену версію фіксованої довжини інших числових даних. Так, файли на декілька гігабайтів та одне слово, будуть перетворюватися в однакову кількість байтів.

Рисунок 2.1 – Алгоритм хешування

Обчислювальні хеш-функції є швидшими за симетричне шифрування. Проте, різниця між ними у тому, що перший – односторонній криптографічний

алгоритм, що є не оборотним. Це пояснюється тим, що не відома 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 вони отримали у перевірках цілісності інформації, алгоритм генерує контрольну суму у файлах, у електронних підписах та зберігання паролів. Але немає гарантій щодо оригінальності файлу, адже зловмисники, замість того щоб змінити частину тексту, можуть замінити все.

2.4. Види алгоритмів шифрування інформації

Захищені обчислювальні середовища враховують різні технології шифрування. Основний розподіл видів полягає у способі використання ключа [9]:

- симетричний – алгоритм при якому секретний ключ, який використовується при шифруванні та дешифруванні інформації однаковий (рис. 2.2);
- асиметричний – використовує різні ключі для шифрування та дешифрування, які не можна отримати один з одного (рис. 2.3).

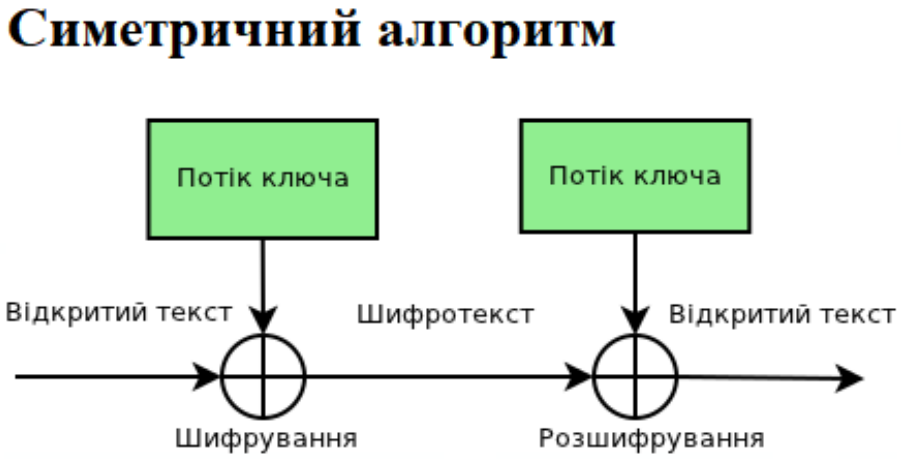


Рисунок 2.2 – Схема симетричних алгоритмів

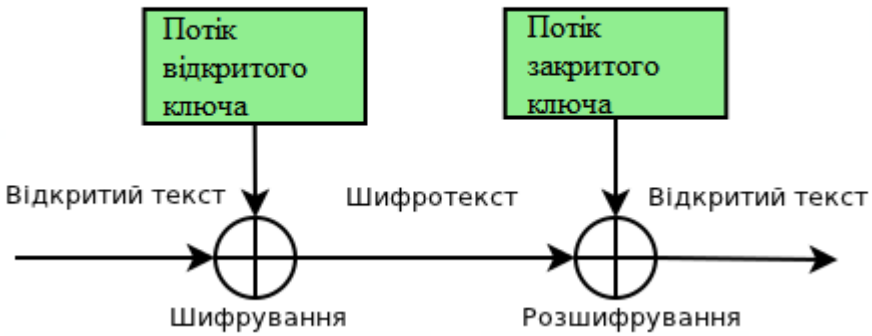


Рисунок 2.3 – Схема асиметричних алгоритмів

Симетричні у свою чергу діляться на:

- потокові;
- блочні.

Потокові – шифрують та дешифрують кожний окремий біт (рис. 2.4). У порівнянні з блочним даний метод є більш складним, але швидшим, оскільки не витрачає час на буферизацію даних з вводу. За основу бере методи зміщення як шифр Цезаря, Плейфера, Гілла, підстановочний та моноалфавітний. Використовують ключ лише один раз. Такі шифри знайшли своє застосування в апаратному забезпеченні, безпечному з'єднанні SSL для Інтернету. Приклади алгоритмів: CFB, OFB.

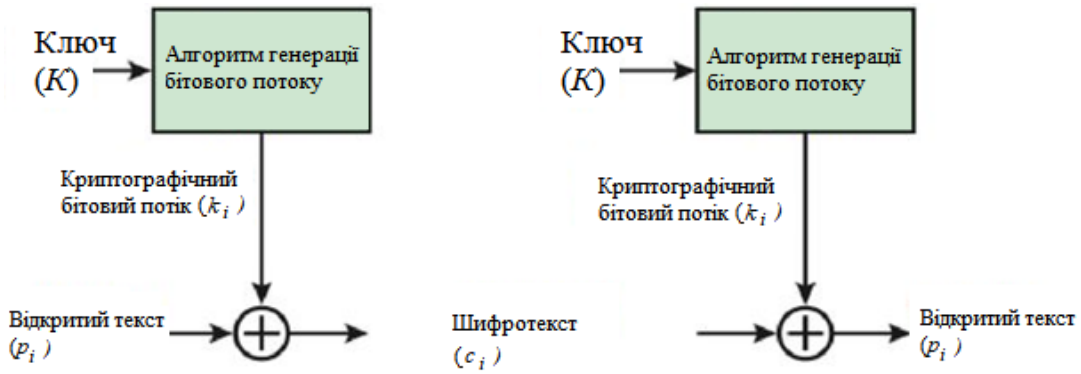


Рисунок 2.4 – Потокове шифрування

Блочні – приймають певну кількість байт або блоків, та працюють з ними перетворюючи в єдину одиницю нової інформації (рис. 2.5). За основу бере методи транспортування як шифр Вернама, перестановочний та книжковий. 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. Використовує 64 або більше бітів, простий, але повільний. Застосовується у програмному забезпеченні, при кодуванні баз даних та файлів. Приклади алгоритмів: ECB, CBC.

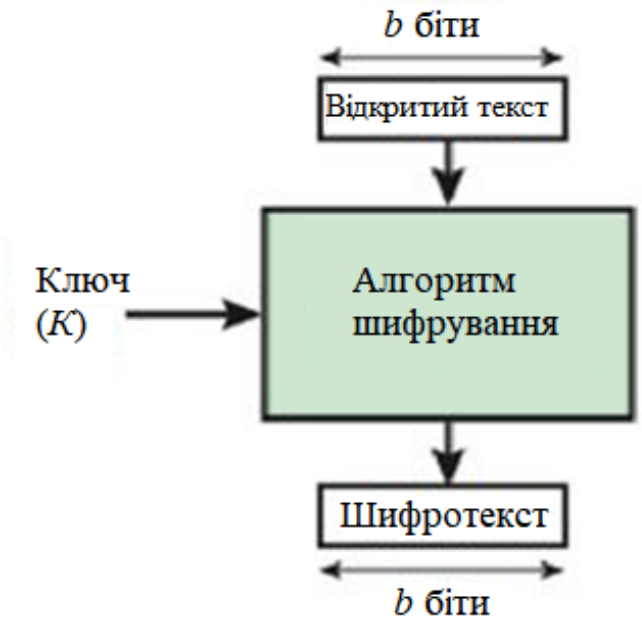


Рисунок 2.5 – Блочне шифрування

Симетричні алгоритми застосовуються для шифрування початкового тексту, наприклад даних на жорстких дисках. Найбільш поширеним був алгоритм – DES на зміну якого, прийшов – AES. При використанні AES з довжиною ключа 256 біт, супер комп'ютер з ресурсами 10 петафлопс на 2008 рік, затратив би близько мільярда років аби зламати це шифрування.

Головним плюсом є те, що він має хороші показники швидкості для читання та запису інформації.

Проте, є й проблеми у даного методу:

- ключі шифрування не є набором тексту, що зрозумілий людям, а набір згенерованих даних, які мають надсилатися абсолютно секретно: фізичним способом або ж не використовуючи мережу;
- при отриманні ключа, весь текст, що передається може бути розшифрованим. У випадку використання асиметричного методу той, хто отримав приватний ключ, може розшифрувати повідомлення, яке надіслали вам, але не може розшифрувати те, що надсилаєте ви.

Для засекречення інформації асиметричні використовують загальнодоступний, відкритий ключ (рис. 2.6), який може отримати будь-хто. Такі ключі застосовуються для шифрування повідомлень особою, яка хоче надіслати інформацію іншій, яка має доступ до цього ключа. Для дешифрації – приватний, що впроваджується для встановлення захищеного з'єднання. Регенерація ключа відбувається лише за умови, якщо буде порушена процедура передачі приватного ключа.

```
04 CE D7 61 49 49 FD 4B 35 8B 1B 86 BC A3 C5 BC D8 20 6E 31 17 2D 92 8A
B7 34 F4 DB 11 70 4E 49 16 61 FC AE FA 7F BA 6F 0C 05 53 74 C6 79 7F 81
12 8A F7 E2 5E 6C F5 FA 10 69 6B 67 D9 D5 96 51 B0
```

Рисунок 2.6 – Відкритий криптографічний ключ

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.

Недоліки використання асиметричних алгоритмів:

- не підходить для кодування великих об’ємів інформації через повільність роботи;
- необхідно перевіряти, чи дійсно вказана особа є власником відкритого ключа;
- при втраті приватного ключа, ніхто не зможе дешифрувати повідомлення.

Швидкість виконання симетричних алгоритмів більша, але у реальному житті, ці два методи інтегруються разом, аби покращити безпеку систем від небажаного витікання інформації, та забезпечення зберігання ресурсів для криптографії.

2.5. Генератори випадкових чисел

Випадкова величина – змінна, значення якої не можна передбачити та послідовність з чисел, які не можна описати жодною формулою, тобто має абсолютну інформаційну ентропію (5). Цей термін для криптографії ввів Шеннон у своїй праці «Математична теорія зв’язку».

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) , \quad (5)$$

де X – випадкова величина;

x_i – можливі значення;

p_i – ймовірності події.

Випадкові величини діляться на [10]:

- дискретні величини:
  - ймовірності між 0 та 1, сума яких дорівнює 1;

- кількість можливих значень можна пронумерувати, тобто кількість або скінченна, або нескінченна зліченна;
- отримання даних шляхом підрахунку.

- неперервні:
  - приймають всі значення на заданому скінченному чи нескінченному інтервалі;
  - розподіл ймовірностей описується кривою щільності;
  - нескінченна кількість можливих значень;
  - отримання даних шляхом вимірювання.

Математичне сподівання – одне з основних понять у теорії ймовірностей, яке обчислюється множенням кожного з можливих результатів на їх ймовірності, середнє значення при повторені одного експерименту.

Математичне сподівання для дискретної випадкової величини.

$$E(X) = \sum_{i=1}^{\infty} p_i x_i \,, \tag{6}$$

де X – випадкова величина;

x_i – можливі значення;

p_i – ймовірності події.

Математичне сподівання для неперервної випадкової величини.

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx \,, \tag{7}$$

де X – випадкова величина;

$f(x)$ – густина розподілу.

Ідеальний генератор випадкових чисел на послідовності (0; 1) видає числа з рівномірним розподілом. За кожну спробу отримати значення, на виході

отримуємо лише одне випадкове число, тобто ймовірності зустріти будь-яке число – однакові.

Оскільки комп'ютери використовують алгоритмічні методи формування послідовностей випадкових значень, то вони залежні від прописаних установ, що робить таку генерацію псевдовипадковою або квазі-випадковою – мають низьку ентропію. Найбільшим плюсом цього методу є те, що вони використовують мінімум ресурсів.

Для визначення істинної генерації випадкових чисел, розглянемо дві послідовності:

1. 10101010
2. 17609246

Оцінюючи задані набори даних, можна зробити висновок, що у першій можна визначити наступне значення даного алгоритму, у другому випадку це зробити складніше, тому існують тести, які можуть дати точне значення наскільки задана послідовність випадкова: Колмогоровська складність.

Колмогоровська складність – це міра обчислювальної здатності, яка необхідна для точного визначення певного об'єкту, наприклад тексту [11].

$$K_f(s) = \min\{|p| : f(p) = s\}, \quad (8)$$

де s – рядок з даними;

f – Тюрінгова машина.

Проблемою даної оцінки є те, що таку функцію K_f не можливо 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. Дану тезу можна пояснити доведенням від супротивного. Наприклад, при написанні програми для підрахунку $K_f(s)$ для $\forall\, s$, необхідно перебрати всі можливі версії реалізації функції, порівнюючи з початковим значенням рядку. Якщо результат співпадає, то повертається

довжина даної конструкції. Але при такому переборі, можливі варіанти нескінченного великої програми, що унеможливить порівняння. Якби це існувало, то для всіх нескінченно можливих скінчених рядків, можна було б згенерувати кінцеве число програм зі складністю нижче n біт. Це схоже на Парадокс Беррі – найменше натуральне число, означення якого неможливо вкласти у задане число слів.

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, nonce та алгоритмах. На разі, використовуються CSPNRG, які приймають на вхід початкове випадкове значення (seed), а згодом підвищують формування неочікуваних послідовностей.

2.6. Дискретний рівномірний розподіл

У криптографії при криптоаналізі одним з основних методів є застосування частот. Для унеможливлення даного варіанту злому алгоритмів існує дискретний рівномірний розподіл (рис. 2.7) – значення коли кінцева кількість результатів виконання процесу мають однакову ймовірність [12]. Прикладом є гральний кубик – ймовірності випадіння кожної сторони становить $\frac{1}{6}$.

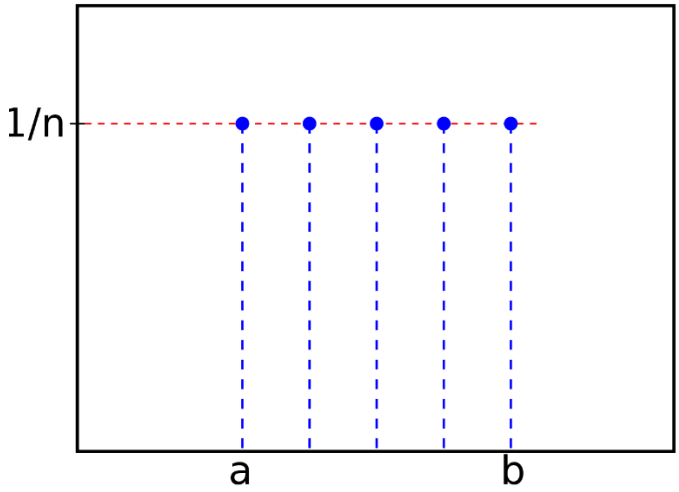


Рисунок 2.7 – Дискретна рівномірна функція маси ймовірностей

Формула рівномірного дискретного розподілу:

$$f(x_i) = \frac{1}{n}, \tag{9}$$

де n – кількість випробувань;

x – значення.

Цього важко досягнути у детермінованих обчислювальних системах, тому можна застосувати метод Alias [13], який працює на двох таблицях: ймовірностей та псевдонімів. Використовується для даних, які мають не змінну таблицю псевдонімів.

Розглянемо приклад (рис. 2.8). Після виконання методу Alias був побудований розподіл Alias-формату, при якому площа прямокутників не змінилася, але шанс потрапити на значення 100 відсотковий, оскільки прогалин на розподілі не залишилося. Тепер кожний елемент складається з двох частин: залишку вихідної та донорського доповнення. Якщо перенести на генерацію випадкових чисел, то шукане значення буде гарантовано отримане.

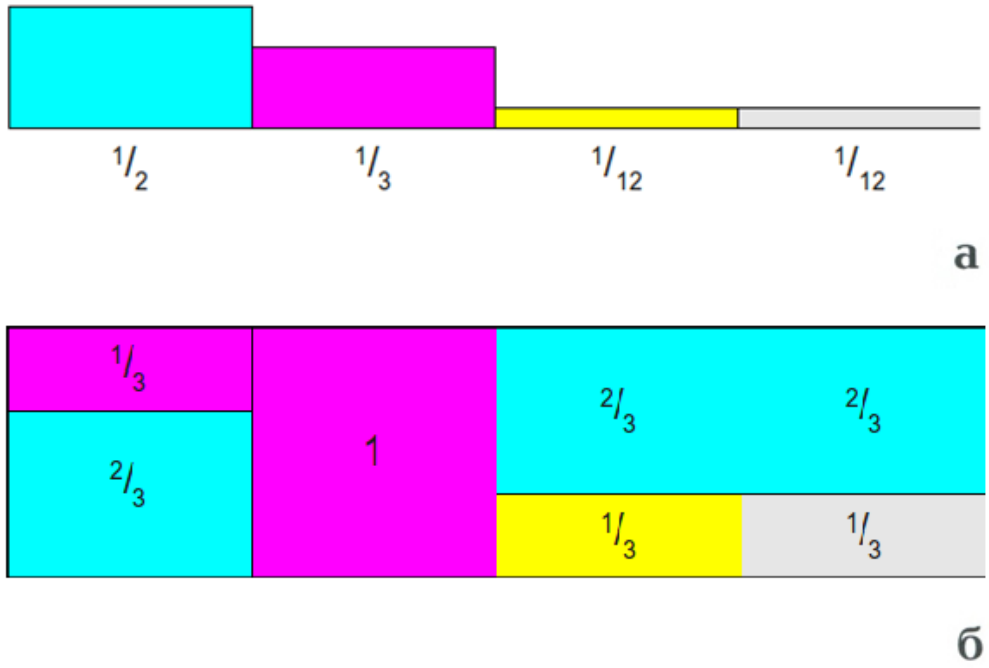


Рисунок 2.8 – Розподіл ймовірностей: а – вхідні значення; б – після застосування методу псевдонімів

2.7. Висновки

У даному розділі проаналізовано, що таке криптостійка система, основні принципи створення алгоритмів для збереження захищеності даних, методи злому та тестування даних, які базуються на застосуванні статистики, математичних парадоксів, вразливостей фізичної складової та інше. Розглянуто плюси та мінуси різних видів шифрування для застосування у даному проєкті та його вдосконалення.

РОЗДІЛ 3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ШИФРУВАННЯ

3.1. Опис алгоритмів та понять, використаних при розробці

У даному проєкті використано Salsa20 для шифрування повідомлення з формуванням ключа за допомогою Вихора Мерсенна та реалізацією ініціалізаційного вектора із застосуванням криптостійкого алгоритму Блум-Блум-Шуба. Для перевірки на цілісність даних – алгоритм хешування SHA-512.

3.1.1. Опис Salsa20

Сучасний алгоритм потокового шифру, який був спроєктований Деніелом Бернштейном, який був представлений на проєкті «eSTREAM». Особливістю потокового методу є те, що кодування та декодування виконуються однією й тою ж самою функцією.

Швидкість алгоритму більше, ніж у RC4 за рахунок того, що виконання кожної ітерації здійснюється за скінченну кількість часу, що дає можливість захиститися від атак по часу (аналіз, скільки система витрачає для завершення роботи криптографічних алгоритмів для різного набору шифротекстів чи ключів). Має гарні показники у застосуванні для у програмному та апаратному забезпеченні. Для процесорів з архітектурою x86 реалізувати метод досить просто, оскільки ця система містить увесь набір команд (інструкцій SSE2), який необхідний для розробки даного алгоритму.

Для реалізації даного алгоритму варто створювати ключ (KEY) – довжиною 128 біт (16 байт) або 256 біт (32 байти) та вектора 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 $2^{64} - 1$ [14].

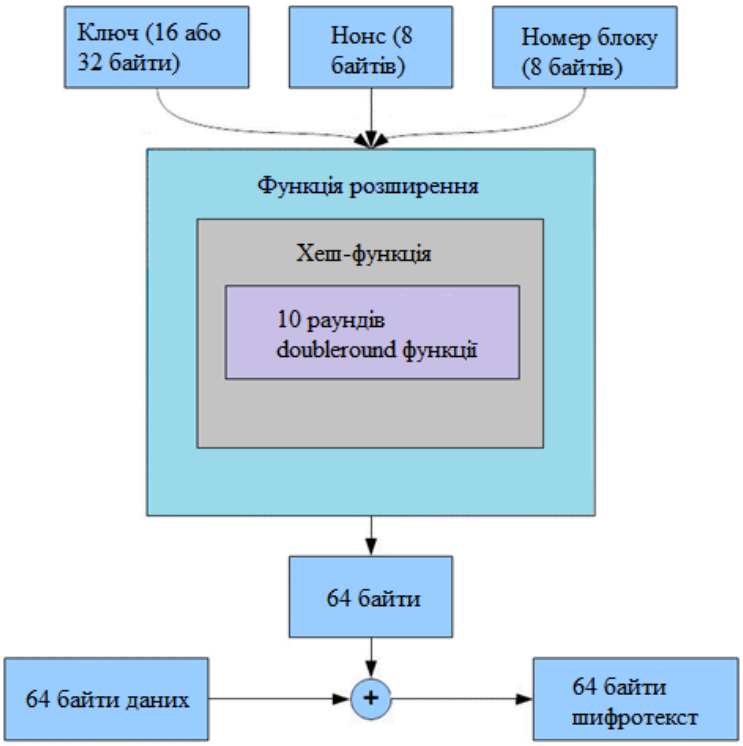


Рисунок 3.1 – Блок-схема алгоритму Salsa20

У Salsa20 застосовуються три прості операції: виключне або (побітове додавання), бітовий циклічний зсув вліво та операція додавання (рис. 3.2).

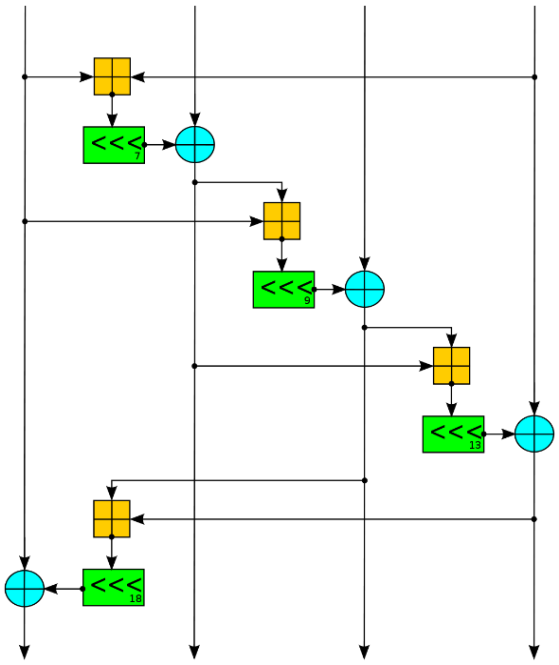


Рисунок 3.2 – Цикл роботи алгоритму Salsa20 із застосуванням операцій

Потокові шифри зручні у випадку даних у вигляді потоків, при якому пакет, які отримав метод, шифрує їх та передає далі. Проте, якщо є набір даних і необхідно зашифрувати лише останні байти, то потрібно застосувати алгоритм до всієї інформації і тоді вже використовувати шифротекст. Дана проблема вирішена у Salsa20 тим, що у систему можна передати розмір даних, які будуть кодуватися.

Недоліком Salsa20 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, тому, для більш важливих даних у комерційних цілях, при роботі з банками та іншому, необхідне автентифіковане шифрування.

Автор алгоритму створив новий варіант XSalsa20, що розширює ключ до 192 бітів [15].

3.1.2. Вихор Мерсенна

Достатньо криптостійкий генератор псевдовипадкових чисел з періодом повторення деякого числа $2^{19937} - 1$, що працює ефективніше приблизно у двадцять разів відносно апаратного RDRand [16]. 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. 3.3).

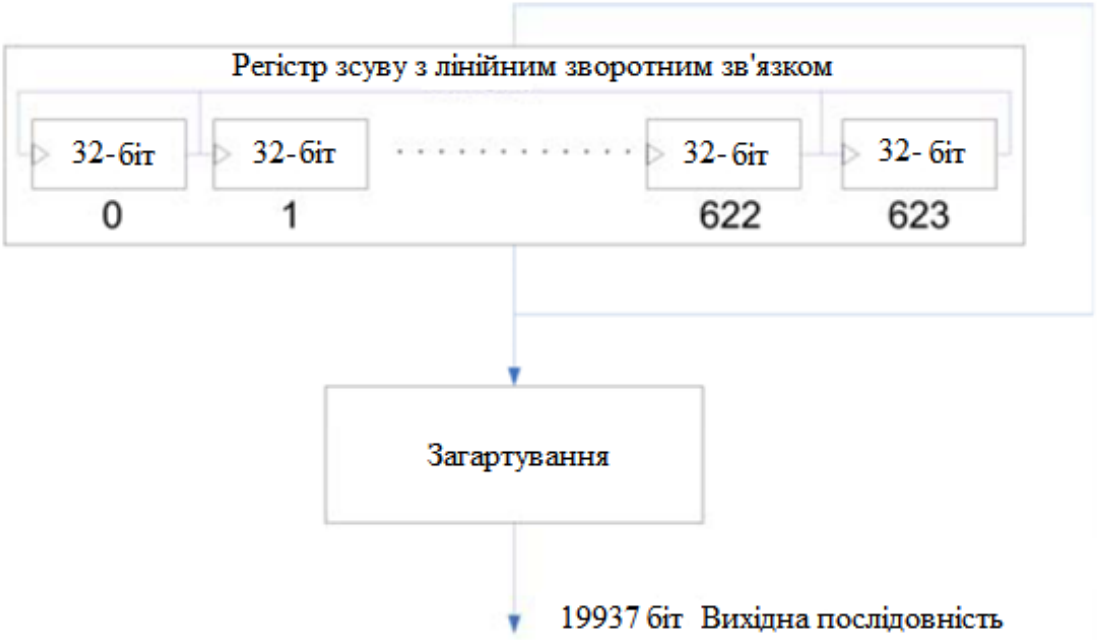


Рисунок 3.3 – Схема Вихора Мерсенна

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:

$$M_n = 2^n - 1\,, \tag{10}$$

де n – натуральне число.

Ця послідовність проходить тести Diehard – 15 тестів, що застосовують статистику для оцінки генераторів псевдовипадкових чисел. На вхід PRNG має подаватися кілька мільйонів 32-розрядних цілих чисел. На виході має бути отримано значення, яке рівномірно розподілене між 0 та 1 [17]:

- парадокс днів народження – відстані між випадково вибраними точками мають мати Пуассонівський розподіл;

- теорема про нескінченних мавп. Якщо у нас є нескінченна кількість мавп, що натискають на клавіші друкарської машинки, то знайдеться певна, що змогла б написати будь-який текст;
- тест на стиснення – вибирається випадкове число на діапазоні $[0, 1)$ та множиться на 2^{31}, поки не отримаємо 1;
- підраховуються одиниці і переводяться в букви, кожні п’ять рахуються випадки п’ятибуквених слів;
- вираховуються 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, що перетинаються – перестановки з п’яти чисел мають бути рівномірно розподілені;
- та інші.

Для застосування даного методу 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 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 Вихором, то він може відновити внутрішній стан регістрів та передбачати значення з стовідсотковою ймовірністю.

У проєкті mt19937 був реалізований за допомогою бібліотеки <random> у просторі імен std [18]. Проініціалізований числом, яке згенеровано 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 дати початкову константу для генератора, який вимагає великої ентропії.

3.1.3. Алгоритм Блум-Блум-Шуба

Для генерації вектору ініціалізації використано алгоритм Блум-Блум-Шуба, оскільки від нього залежить, як будуть шифруватися однакові повідомлення [19]. На відміну від Вихора Мерсенна, даний метод набагато повільніший, але надійніший, що забезпечується створенням значень шляхом факторизації чисел:

$$x_{n+1} = x_n^2 \, mod \, (p * q) \, , \tag{10}$$

де p, q – великі прості числа;

x_{n+1} – початкове значення для кожної наступної ітерації;

x_n – вихідні дані.

З кожним кроком, алгоритм формує вихідні дані 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, та береться останній біт значення x_n.

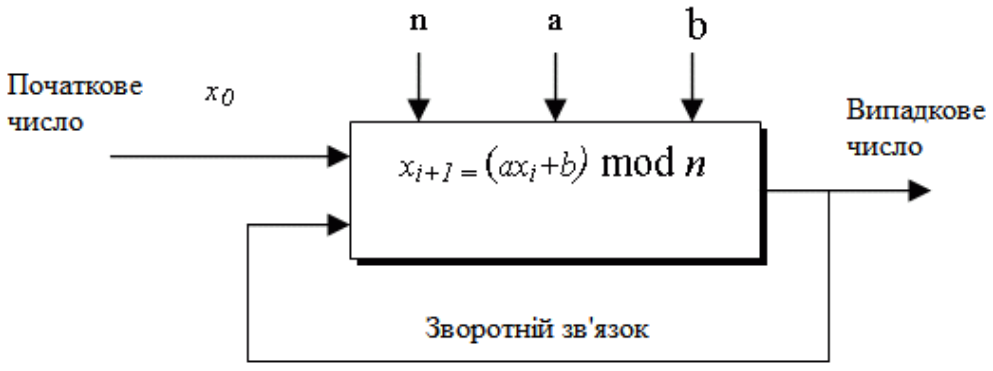


Рисунок 3.4 – Схема Блум-Блум-Шуба

Для того, аби зламати цю систему, необхідно знати початкові вхідні дані – p, q або до тих пір, поки не буде винайдено швидкий алгоритм факторизації чисел.

3.1.4. Алгоритм хешування SHA-512

SHA – безпечний алгоритм, який використовується для створення хеш-суми (дайджест) вхідних даних для встановлення цілісності повідомлення чи ідентифікації, що базується на хеш-функції Меркла-Демґарда (рис. 3.5). Спочатку функція застосовує доповнення для створення блоків, які кратні 512, розбиває вхідні дані і до кожного блоку застосовує алгоритм стискання [20].

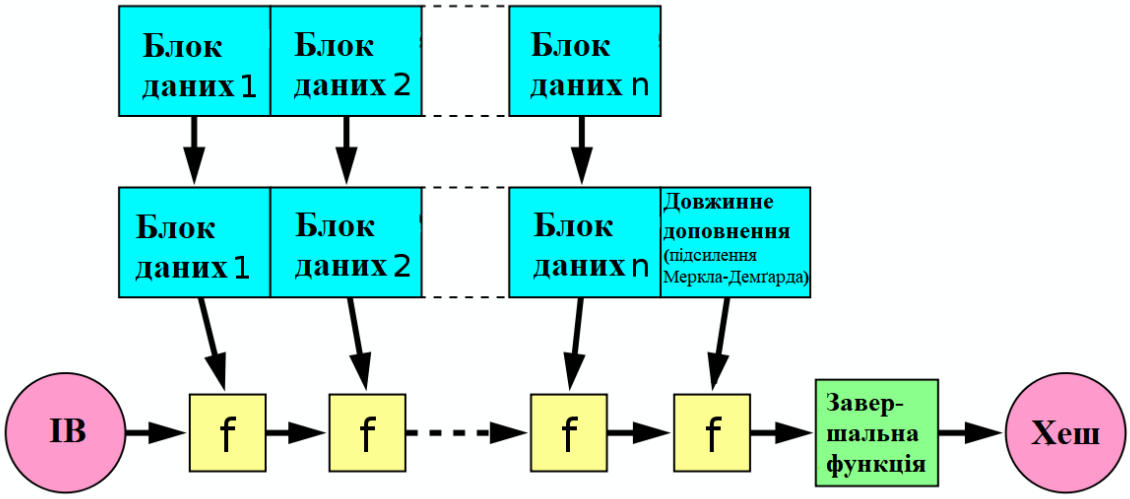


Рисунок 3.5 – Будова Меркла-Демґарда для хешування

Алгоритм SHA-512 використовує ініціалізаційний вектор, завершальну функцію (застосовує стискання, змішування або ж лавиновий ефект) і стискання (поєднується з наступним блоком повідомлення) [21]. Створює 1024 бітні блоки і здатен приймати довжину рядка 2^{128} біт, після застосування, 80 раундів видають 64 байти хешу, початкова інформація інтерпретується у вигляді 128 шістнадцяткових цифр (рис. 3.6). У раундах застосовуються константи – перші 80 простих чисел.

Проте, навіть такі функції піддаються атакам, через відшукання колізій. Так, китайці для HAVAL, RIPEMD та MD5 знайшли вразливості, через які ваш комп’ютер може зламати хеші за 15 секунд.

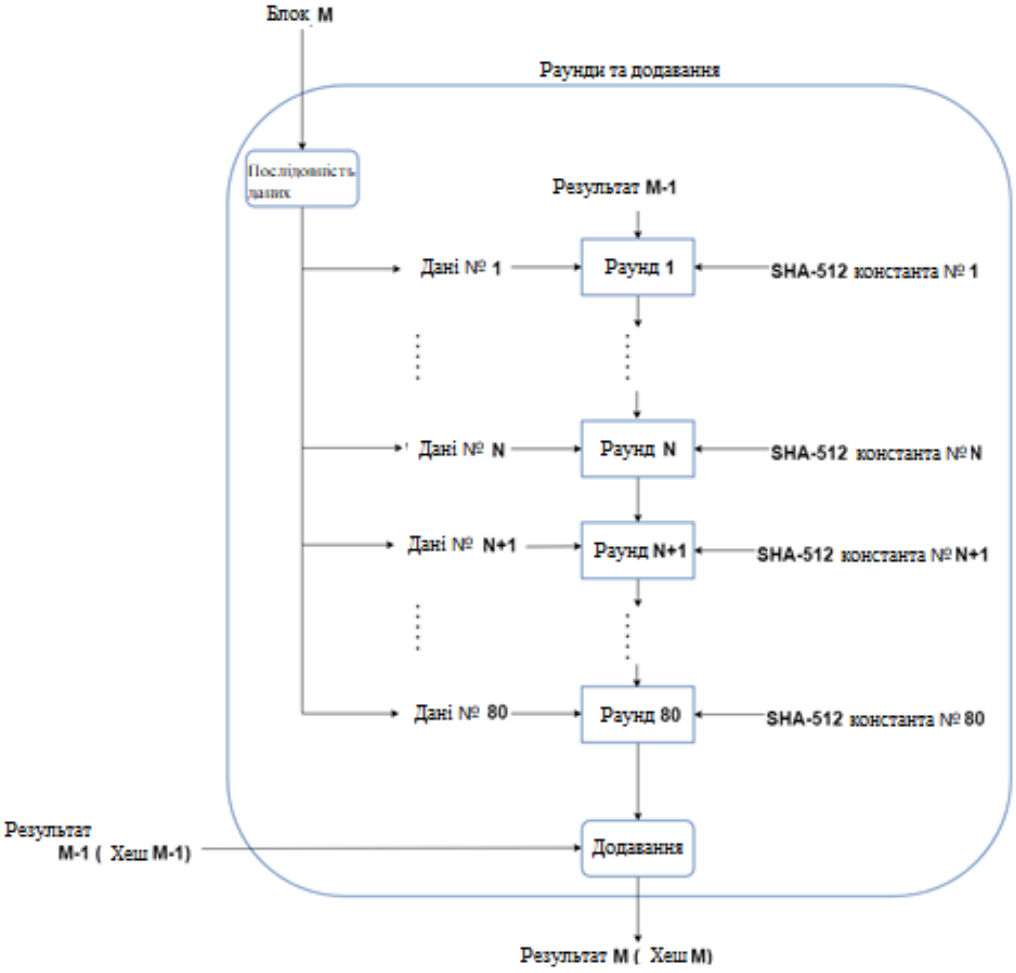


Рисунок 3.6 – Алгоритм SHA-512

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 512 версії.

3.2. Вибір технологій для програмної реалізації

Для реалізації проєкту була використана мова програмування C++ для роботи з пам’яттю низького рівня та програмне забезпечення QT для створення графічних інтерфейсів користувача (GUI). Даний вибір забезпечує можливість використати одну мову та фреймворк без додаткових інструментів для вирішення практично будь-яких завдань.

C++ – мова, на якій можна писати швидкі, прості, а головне компактні програми. Оскільки мова низькорівнева, на ній можна писати різні програми – від драйверів до ігор. Ця мова надає розробникам повний контроль над управлінням пам’яттю (рис. 3.7), що підходить для концепції розробки алгоритмів шифрування, які працюють з бітами та операціями над ними [22].

Рисунок 3.7 – Розподіл пам’яті у C++

Проте, з цим постає ряд недоліків, що зобов’язує програмістів слідкувати за витоками пам’яті та співпрацювати з вказівниками. Транслювання у машинну мову надає змогу аналізувати код Assembler для виправлення низькорівнених помилок. Також, є можливість робити відповідні вставки у середовище розробки для роботи з регістрами. Це одна з найстаріших мов, тому на сьогодні, вона має велику кількість бібліотек з алгоритмами, структурами, функціями та інше. Для шифрування даних можна використати Crypto++ –

бібліотеку, яка містить криптографічні класи з реалізацією різних алгоритмів шифрування та псевдогенераторів чисел.

QT – IDE платформа, що підтримує компілятори GCC, G++, Visual Studio, дає змогу відлагоджувати код, може застосовуватися на будь-якій операційній системі завдяки кросплатформеності, має добре документовану структуру, містить ООП (об'єктно-орієнтоване програмування), що дозволяє наслідувати код [23].

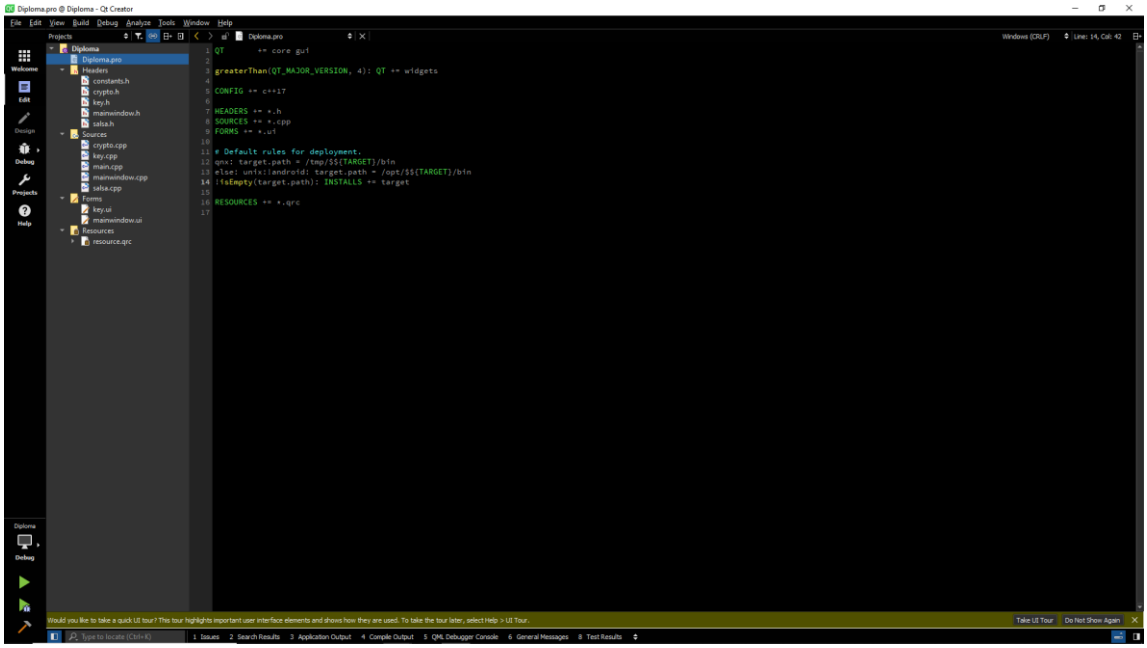


Рисунок 3.7 – Програмне забезпечення QT Creator

Для зв’язку між реалізацією інтерфейсу для користувача та коду, віджети надсилають сигнали (слоти), що містять інформацію про подію. Може використовувати cmake, qmake для автоматизації генерації Makefiles.

Недоліки QT:

- візуальна частина для iOS та Android може набувати не ідеального вигляду, яку не можливо змінити (у даній роботі це не критично);
- при створенні потоків і застосуванні основних бібліотек QObject та QWidget можливі помилки;

- важкість у розумінні помилок на стеку (динамічна структура даних з принципом Last-In First-Out);
- складне налагодження роботи з пам’яттю, її витоками.

Для контролю версій програми використано GIT, що дозволяє візуально прослідкувати зміни та швидке злиття гілок. Працює з даними за допомогою снапшотів – у певний момент часу робить копію (знімок) файлової системи та зберігає посилання на них. Уся історія проєкту зберігається локально на диску пристрою, тому швидкість роботи даної системи – миттєва. Кожного разу при внесенні змін GIT просто додає дані до своєї стиснутої бази даних.

3.3. Опис архітектури системи

Система складається з .ui (user interface – реалізація візуальної частини) файлів, .h (заголовні файли – декларація змінних, класів та функцій) та .cpp (вихідний код на мові C++). Для роботи з байтами використано стандартну бібліотеку C++ – cstdint та QT – QByteArray.

Модулі програми:

- mainwindow.ui, mainwindow.cpp, mainwindow.h – реалізація головного вікна для вибору та виконання кодування даних, відкриття вікна з ключем, виходу з програми;
- salsa.cpp, salsa.h – реалізація алгоритму Salsa20/20;
- key.ui, key.cpp, key.h – створення та вивід ключа та ініцілізаційного вектору на вікно «Ключ», які реалізуються за допомогою Вихора Мерсенна, який написаний у стандартній бібліотеці random та з використанням Блум-Блум-Шуба;
- 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 Salsa20.

Для запуску програми на персональному комп’ютері необхідне програмне забезпечення QT з стандартними бібліотеками, у іншому випадку необхідно скачати відповідні заголовні файли і додати їх у папку з розширенням .exe: Qt5Core.dll, Qt5Gui.dll, Qt5Widgets.dll, libgcc_s_seh-1.dll, libstdc++-6.dll, qt.conf. Потім потрібно створити папки: \plugins\platforms у які вставити qminimal.dll та qwindows.dll.

3.4. Опис інтерфейсу системи

Роботу системи протестовано на Windows 10 Pro. Після запуску .exe програми відкривається головне вікно (рис. 3.7).

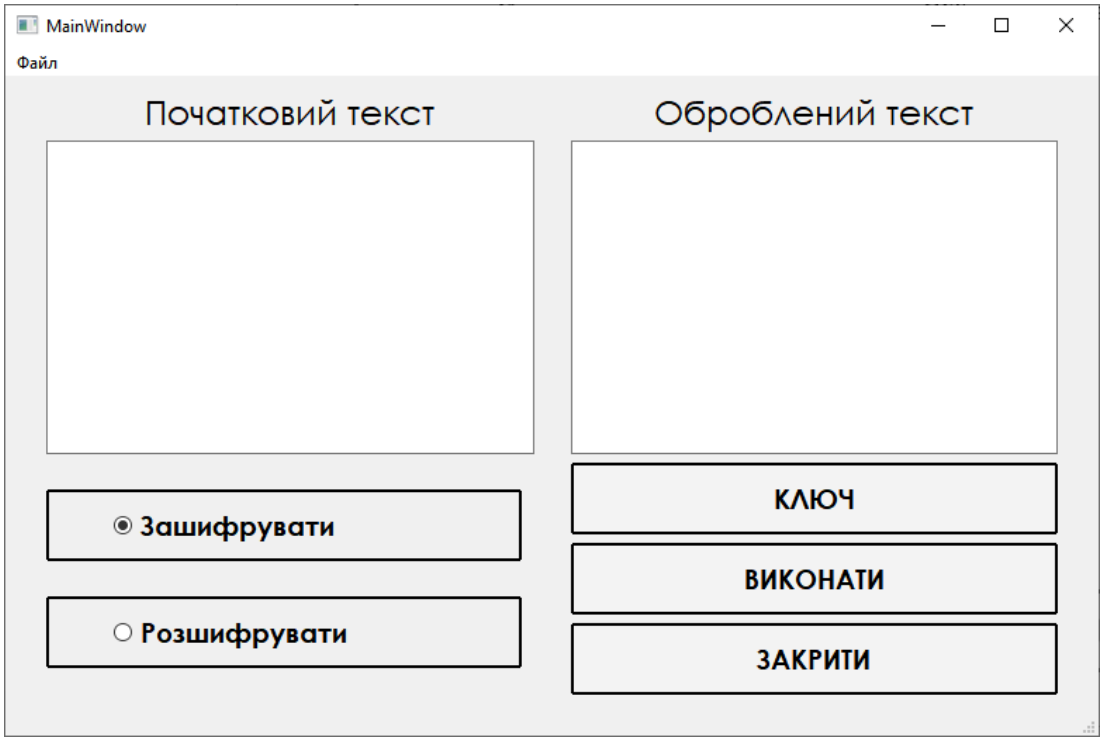


Рисунок 3.8 – Головне вікно програми

Для того аби зашифрувати повідомлення, необхідно у «Початковий текст» ввести дані та вибрати варіант «Зашифрувати». При цьому спочатку необхідно натиснути на «Ключ», після чого відкриється нове вікно, у якому відбувається генерація ключа та ініцілізаційного вектору (рис. 3.8).

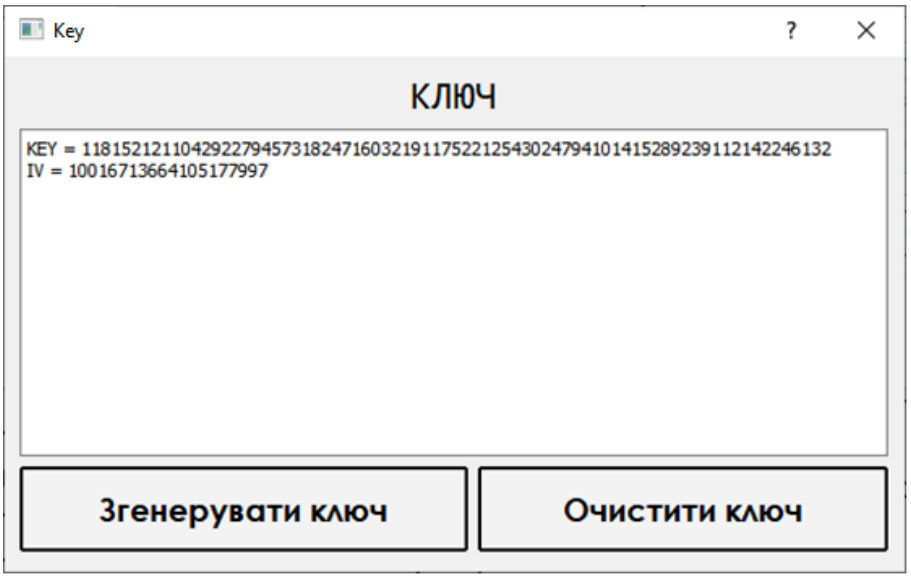


Рисунок 3.9 – Згенерований ключ та ініціалізаційний вектор

Після закриття вікна «Key», відбувається перехід на основне меню. При натисканні на кнопку «Виконати» зашифровані дані за допомогою Salsa20 з’являться у «Оброблений текст» (рис. 3.9).

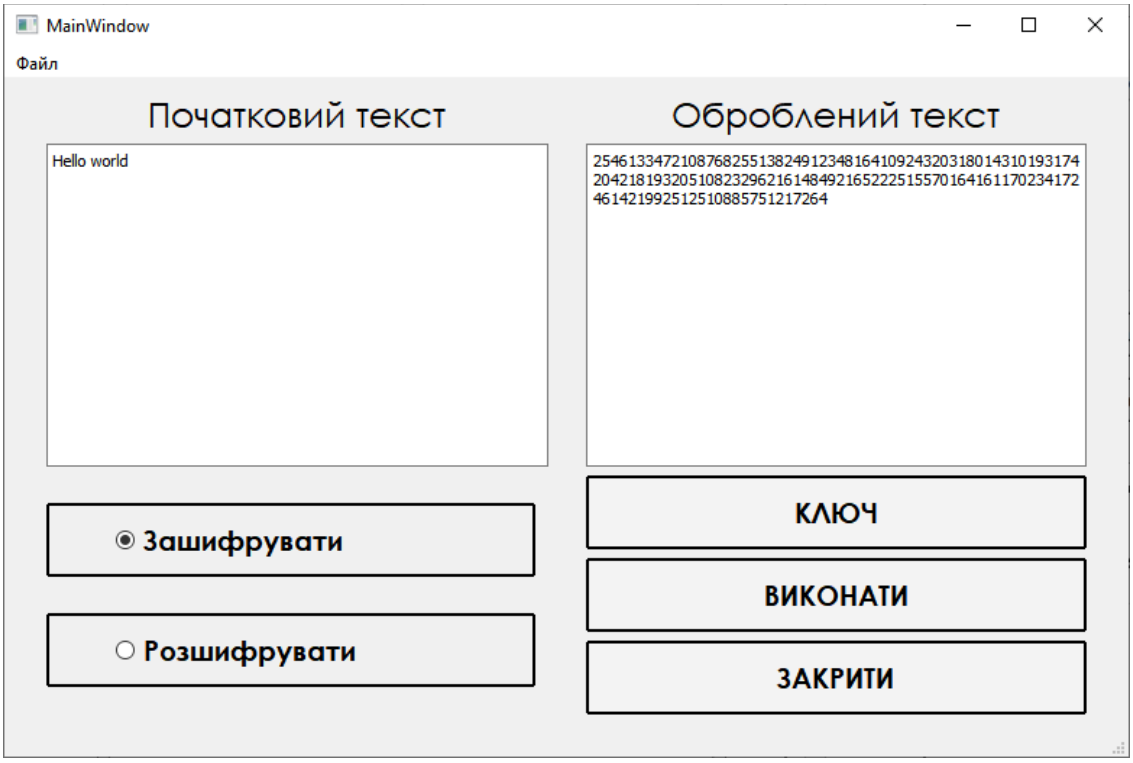


Рисунок 3.10 – Приклад шифрування даних системою

У випадку якщо ключ не згенерований, при намаганні шифрувати або розшифровувати дані, з’явиться попередження (рис. 3.10).

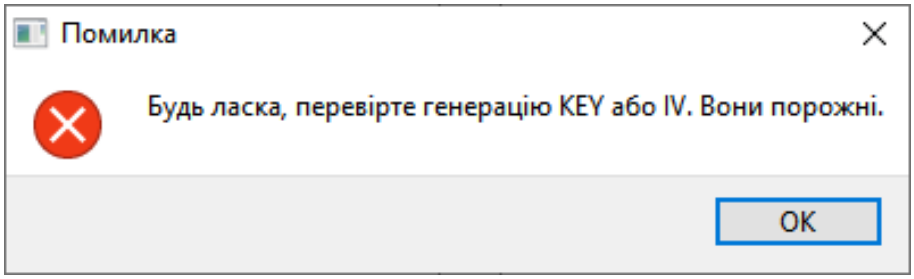


Рисунок 3.11 – Помилка пустих даних

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. 3.11).

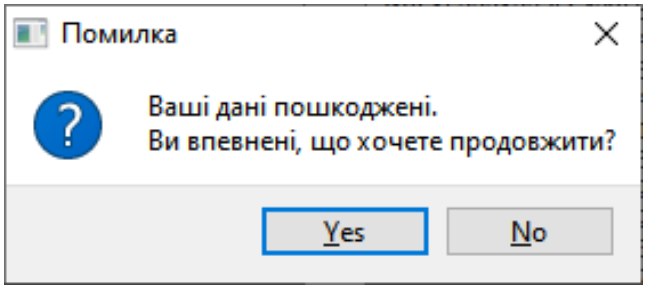


Рисунок 3.12 – Помилка цілісності даних

При натисканні «Yes», система розшифровуватися пошкоджені дані, що призводить до неправильних вихідних даних (рис. 3.12).

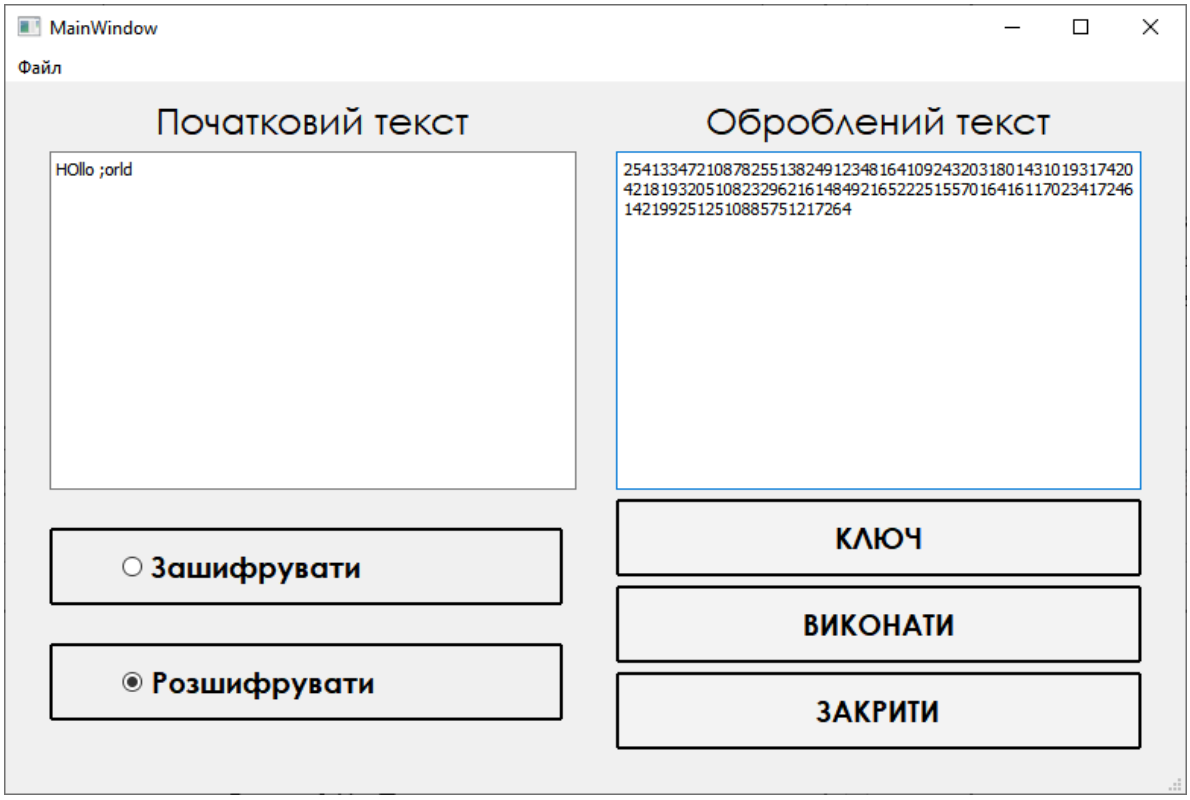


Рисунок 3.13 – Приклад пошкоджених даних

3.5. Пропозиції для покращення системи

Можливості, які можна впровадити для покращення системи шифрування:

- для збільшення функціональної частини, можна впровадити залиття файлів у систему та їх вигрузку;
- вибір розміру блоків для шифрування даних;
- застосування алгоритмів шифрування для передачі ключа та ініцілізаційного вектору, збільшення їх розмірів, надійне зберігання;
- можливе використання генераторів випадкових чисел, які будуть залежати від величин, які не можливо передбачити (мінливість фізичних властивостей – температура процесора, рух квантів та інше.);

- збільшення циклів проходження кодування.

Дана система представлена лише у вигляді моделі для реалізації алгоритмів шифрування даних в якій можна впроваджувати нові технології для застосування у мережевих цілях.

3.6. Висновки

Даний розділ має характеристику вибору основних алгоритмів шифрування системи, реалізації інтерфейсної частини програми та її архітектури, застосування стандартних бібліотек та фреймворків, технологій, які були використані при розробці програми.

Завдяки вибору технологій C++/QT реалізація системи є простою при роботі з пам’яттю та візуалізацією. Також у розділі висвітлено застосування описаних алгоритмів та усунуто недоліки у системі, яка розроблена для шифрування даних. Відповідно, з розвитком операційних можливостей машин, методи злому вдосконалюються. Але, за допомогою відкритого коду та залучення експертів у галузі криптографії можна покращити дані алгоритми.

ВИСНОВКИ

Під час роботи над даним дипломним проєктом проаналізовано основні алгоритми шифрування даних їх переваги та недоліки, досліджено методи щодо їх злому, обрано швидкі потокові алгоритми кодування, а також псевдогенератори випадкових чисел для реалізації констант та ключів.

У першому та другому розділі викладена основна теоретична частина щодо криптографії та аналіз актуальності проблем, щодо впровадження систем, які здатні зберігати конфіденційність і цілісність інформації. Проаналізувавши теорію, можна сказати, що проблема буде завжди на високому рівні дослідження та вирішення, адже з розвитком обчислювальних можливостей машин зменшуються періоди для встановлення вразливостей та колізій для алгоритмів.

Третій розділ містить опис програми: інструментів, які застосовувалися та файли з яких складається реалізація. Описана система має рекомендації щодо покращення у майбутньому, при реалізації яких, розробку можна буде використовувати в Інтернеті.

Поставлену задачу виконано успішно. У результаті створена програма вдосконаленого шифрування даних із застосуванням уже наявних. Програма має гарну візуальну частину для тестування, а також є універсальною, адже використовує технології, які підтримуються на усіх операційних системах та не вимагають багато пам’яті.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Cryptography : URL: https://ethw.org/Cryptography?gclid=Cj0KCQjwyZmEBhCpARIsALIzmnJTKAN82GBvieXzL7eihj4lUwyNeB4aoN5zi74GgW-ykBNSc1bynzQaAvlbEALw_wcB (дата звернення: 15.04.2021).
2. HOW ALAN TURING CRACKED THE ENIGMA CODE : URL: https://www.iwm.org.uk/history/how-alan-turing-cracked-the-enigma-code – (дата звернення: 15.04.2021).
3. Attacks On Cryptosystems : URL: https://www.tutorialspoint.com/cryptography/attacks_on_cryptosystems.htm (дата звернення: 09.05.2021).
4. Types of Encryption: 5 Encryption Algorithms & How to Choose the Right One : URL: https://www.thesslstore.com/blog/types-of-encryption-encryption-algorithms-how-to-choose-the-right-one/ (дата звернення: 06.05.2021).
5. Разложение числа на простые множители (факторизация). Делители числа : URL: https://brestprog.by/topics/factorization/ (дата звернення: 06.05.2021).
6. Perfect Secrecy : URL: https://www.sciencedirect.com/topics/computer-science/perfect-secrecy – (дата звернення: 15.04.2021).
7. Kerckhoffs's principle : URL: http://www.crypto-it.net/eng/theory/kerckhoffs.html – (дата звернення: 18.04.2021).
8. Authenticated encryption : URL: https://www.cryptopp.com/wiki/Authenticated_Encryption (дата звернення: 02.05.2021).
9. Symmetric And Asymmetric Key Cryptography: All You Need To Know In 3 Points : URL: https://www.jigsawacademy.com/blogs/cyber-

security/symmetric-and-asymmetric-key-cryptography/ – (дата звернення: 21.04.2021).

10. Discrete and Continuous Random Variables : URL: http://www.henry.k12.ga.us/ugh/apstat/chapternotes/7supplement.html – (дата звернення: 19.04.20211).
11. Kolmogorov Complexity : URL: https://people.cs.uchicago.edu/~fortnow/papers/kaikoura.pdf (дата звернення: 03.05.2021).
12. Uniform Distribution : URL: https://corporatefinanceinstitute.com/resources/knowledge/other/uniform-distribution/ (дата звернення: 07.05.2021).
13. Darts, Dice, and Coins: Sampling from a Discrete Distribution : URL: https://www.keithschwarz.com/darts-dice-coins/ (дата звернення: 07.05.2021).
14. The Salsa20 core : URL: https://cr.yp.to/salsa20.html (дата звернення: 03.05.2021).
15. XSalsa20 : URL: https://libsodium.gitbook.io/doc/advanced/stream_ciphers/xsalsa20 (дата звернення: 03.05.2021).
16. How does the Mersenne's Twister work? : URL: https://www.cryptologie.net/article/331/how-does-the-mersennes-twister-work/ (дата звернення: 04.05.2021).
17. The Diehard Tests : URL: http://theurbanengine.com/blog//the-diehard-tests (дата звернення: 05.05.2021).
18. std::mersenne_twister_engine : URL: https://en.cppreference.com/w/cpp/numeric/random/mersenne_twister_engine (дата звернення: 05.05.2021).
19. Blum-Blum-Shub generator и его применение : URL: https://habr.com/ru/post/534698/ (дата звернення: 07.05.2021).

20. Secure Hash Algorithms : URL: https://brilliant.org/wiki/secure-hashing-algorithms/ (дата звернення: 07.05.2021).
21. SHA-512 Hashing Algorithm Overview : URL: https://komodoplatform.com/en/blog/sha-512/ (дата звернення: 08.05.2021).
22. Динамічний розподіл пам’яті у C++ : URL: https://dl.sumdu.edu.ua/textbooks/108989/459248/index.html (дата звернення: 04.05.2021).
23. Cross-platform software development : URL: https://www.qt.io/ (дата звернення: 04.05.2021).