Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра програмних засобів

ЗВІТ

з лабораторної роботи № 2

з дисципліни «Безпека та захист програм і даних» на тему:

«РЕЖИМИ ШИФРУВАННЯ БЛОКОВИХ ШИФРІВ»

Виконав:

ст. гр. КНТ-113сп Іван ЩЕДРОВСЬКИЙ

Прийняв:

доцент Тетяна ЗАЙКО

2025

1 Мета роботи

Ознайомитись з програмною реалізацією алгоритму симетричного блокового шифрування RIJNDAEL.

2 Завдання до лабораторної роботи

2.1 Керуючись довідковими матеріалами та мережею Інтернет, зрозуміти наступні математичні перетворення: "додавання за модулем 2", "дискретне поле Галуа" та "перетворення у дискретних полях Галуа";

2.2 Детально описати алгоритм зворотнього перетворення (дешифрування);

2.3 Зрозуміти та детально описати відмінності алгоритму AES-256;

2.4 Керуючись довідковими матеріалами та мережею Інтернет, познайомитись із блочним симетричним шифром "Калина"(ДСТУ 7624:2014) та порівняти його з AES.

3 Відповіді на контрольні питання

З яких операцій складається раунд алгоритму RIJNDAEL?

Раунд алгоритму складається з чотирьох операцій

Перша це “SubBytes”. При цій операції кожен байт замінюється на інший байт відповідно спеціальної стандартизованої таблиці перетворення

Друга операція це “ShiftRows”. В ній другий рядок циклічно здвигається на 1 елемент вліво, третій на два та четвертий на три

Третя це “MixColumns”. Вона не застосовується на останньому раунді. При цій операції всі данні помножуються на спеціальну стандартизовану матицю

Четверта операція це “AddRoundKey”. При ній ми беремо спеціальний round key, який був створений через розширення початкового ключа та робимо операцію XOR з нашими даними.

Використання “SubBytes” та “AddRoundKey” додає складну залежність між вхідними даними та зашифрованими результатами. Наприклад, якщо взяти XOR, то залежність між ключем та зашифрованими результатами є досить очевидною

Використання ж “ShiftRows” та “MixColumns” додають diffusion, тобто при зміні одного байту вхідних даних буде змінено багато байтів вихідних даних.

Поясніть режим електронна кодова книга (ECB).

Electronic Codebook (ECB) mode, або ж режим кодової книги – це режим роботи алгоритмів шифрування при якому кожен блок відкритого тексту зашифровується в один блок зашифрованого тексту без будь-якого зв’язку між блоками

Тобто, весь текст розбивається на блоки, всі блоки шифруються окремо та потім об’єднуються в один великий текст.

Але, наприклад, ми отримуємо проблему, що два однакових блоки відкритого тексту будуть створювати два однакових блоки шифротексту, що не дуже безпечно

Поясніть режим зчеплення блоків шифротексту (CBC).

Cipher Block Chaining, або ж режим зчеплення блоків – це режим при якому відкритий текст кожного блока шифрується через XOR з результатом попереднього блоку перед основним алгоритмом шифруванням. Для шифрування першого блоку використовує Initialization vector.

Таким чином два однакових блоки відкритого тексту вже не будуть відповідати двом однаковим блокам шифротексту, оскільки є ось цей XOR з попереднім блоком

Поясніть режим зворотнього зв’язку за шифротекстом (CFB).

Cipher feedback, або режим зворотнього зв’язку за шифротекстом – це режим при якому шифрування відбувається не з текстом, а з шифротекстом, або ж Initialization vector. А відкриті данні після шифрування будуть додані через XOR

Це означає цікаву річ – для початку алгоритму шифрування нам, так само як в CBC, потрібно мати весь шифротекст з попереднього блоку, але при цьому не потрібно мати весь відкритий текст, як це потрібно в CBC!

Тобто ми можемо одночасно читати інформацію з файла та одночасно її шифрувати, не чекаючи поки ми її прочитаємо всю. Саме тому цей режим застосовується для потоку даних

Але все одно ми не можемо кодувати блок 1 та 2 одночасно, бо для блоку 2 потрібно мати весь шифротекст блоку 1!

Поясніть режим зворотнього зв’язку по виходу (OFB).

Output feedback, або ж режим зворотнього зв’язку по виходу – це режим в якому для шифрування через основний блоковий алгоритм взагалі не використовуються відкриті дані в усіх блоках

В перший блоковий шифр передається ініціалізацій вектор, в другий – результат першого, в третій – результат другого і так далі

А шифрування відкритих даних в шифротекст відбувається через XOR в кожному блоці

Це означає, що ми можемо спочатку знайти результати роботи блокового шифру для всіх потрібних нам блоків, а після цього паралельно робити шифрування кожного окремого блоку з вихідним текстом.

Тобто, ми не можемо паралельно вираховувати результати всіх блокових шифрів, бо вони залежать один від одного, але при цьому ми можемо дуже гарно паралельно обчислювати шифртекст.

Поясніть роботу алгоритму в режимі шифрування з лічильником (Counter).

Counter, або ж CTR, або ж режим шифрування з лічильником – це режим шифрування при якому кожен блок шифрується окремо та незалежно від іншого, але для забезпечення безпеки, на відміну від ECB, використовується спеціальний лічильник

Для кодування даних спочатку робиться шифрування nonce, або ж number used once/постійної частини, яка об’єднана з деяким лічильником використовуючи блокове шифрування, а потім робиться XOR з відкритими даними

Лічильник представляє собою функцію яка повертає унікальні значення на довгому проміжку часу. Зазвичай це звичайний інкремент числа. Деякі говорять, що це зменшує безпеку, але блоковий шифр повинен її забезпечувати і так

Використовуючи подібний підхід ми отримуємо відразу два великих плюси. Перший це те, що такий алгоритм можна дуже просто запустити паралельно, оскільки кожен блок є незалежним. А другий – оскільки підмішування вхідних даних відбувається через XOR вже до результату блокового шифру ми можемо використовувати цей режим для роботи в потоці даних!

Чому шифри ГОСТ 28147-89 та RIJNDAEL називаються блоковими шифрами?

Тому-що для їх використання потрібно мати дані деякої довжини. Якщо ми читаємо дані в потоці та хочемо їх шифрувати то нам потрібно дочекатись поки набереться достатня кількість байтів, оскільки ми не можемо робит

Якими можуть бути довжина ключа та блока в ГОСТ 28147-89 та в RIJNDAEL?

ГОСТ 28147-89 має 256 бітовий ключ, 32 цикли перетворення та оперує 64-бітними блоками

RIJNDAEL – має 128, 192 або 256 бітовий ключ та оперує над 128, 192 або 256 блоками. Кількість раундів залежить від ключа та блоку.

Якщо і ключ і блок містять 128 біт, то 10 раундів.

Якщо ключ/блок містить 192 біт, а блок/ключ 128 або 192 біт то 12 раундів.

Якщо ключ або блок містять 256 біт, то 14 раундів не залежно від розміру іншого.

В чому полягає основний шаг криптосистем ГОСТ 28147-89 та RIJNDAEL?

Основа ГОСТ 28147-89 це Мережа Файстеля, Feistel cipher, а ще його називають Luby–Rackoff block cipher

Основою RIJNDAEL є Substitution-Permutation Network

Перелічіть основні режими шифрування блокових шифрів.

ECB, Electronic Codebook
CBC, Cipher Block Chaining
CFB, Cipher Feedback
OFB, Output Feedback
CTR, Counter

Детально всі режими були описані в попередніх запитаннях

Які основні параметри шифрів ГОСТ 28147-89 та RIJNDAEL?

ГОСТ 28147-89 – блочний симетричний шифр, блок 64 біт, ключ 256 біт, має 32 раунди. Працює в режимі простої заміни, в режимі гамування, режимі гамування з зворотним зв’язком та режимі виробітки імітовставки

RIJNDAEL - блочний симетричний шифр, блок 128/192/256 біт, ключ 128/192/256 біт, раундів 10/12/14, де кількість залежить від ключа та блоку. Може працювати в режимах ECB, CBC, CFB, OFB, CTR та інших

У чому полягає режим простої заміни?

Цей режим працює як ECB, який був описаний вище

У чому полягає режим гамування?

Цей режим працює як CTR, який був описаний вище

У чому полягає гамування зі зворотним зв’язком?

Цей режим працює як CFB, який був описаний вище

У чому полягає режим виробітки імітовставки?

MAC, Message authentication code, або ж імітовставка використовується для перевірки цілісності даних, тобто щоб підтвердити, що дані прийшли від відправника та вони не були змінені

Режим виробітки імітовставки працює приблизно так само як CBC-MAC режим

CBC-MAC режим працює як CBC режим, але замість Initialization vector завжди використовується 0, а також результат береться тільки з останнього блоку. Тобто, при розміру блока в 128 біт розмір MAC також буде 128 біт

Для створення MAC потрібно брати >=2 блоків, а сам MAC блок додається в кінець даних. При отриманні результатів виконується така сама операція і перевіряється, що данні не були змінені.

4 Керуючись довідковими матеріалами та мережею Інтернет, зрозуміти наступні математичні перетворення: "додавання за модулем 2", "дискретне поле Галуа" та "перетворення у дискретних полях Галуа"

Операція “додавання за модулем 2”, або ж виключне “АБО”, або ж XOR це логічна операція результат якої дорівнює один якщо вхідні значення різні та 0 якщо вхідні значення однакові.

Також можна сказати, що XOR змінює біти на протилежні в першому вхідному значенні, якщо біт в тому самому положенні в вихідних даних дорівнює 1.

Наприклад, хай перший рядок буде 1001, а другий 1100.

Тепер змінимо значення бітів з першого рядку на протилежні якщо в другому на відповідній позиції біту стоїть 1. Отримаємо 0101, що і є результатом XOR

Також є більш стандартний метод, та причина чому це називається додавання за модулем 2, ми додаємо відповідні біти та знаходимо модуль від двох

1 + 1 = 2 mod 2 = 0. Ось чому це 0. Якщо взяти різні біти, то 1 + 0 = 0 + 1 = 1 mod 2 = 1. Ось чому це 1. Ну і 0 + 0 = 0 mod 2 = 0.

“Поле Гаула”, або ж “Дискретне поле Гаула”, або ж “Скінченне поле” – це поле, яке складається зі скінченної множини елементів

Найменше поле Гаула GF(2) містить лише два елементи: 0 та 1. Поле GF(P) не може містити елементи більші, а ніж P – 1 включно. А всі елементи які додаються будуть множитись “за модулем p”. Наприклад, якщо в полі GF(2) додати 1 + 1 то ми отримаємо 0. Також такі поля можна записати як $GF(p^k)$, наприклад $GF(2^8)$, який буде представляти 8 біт, де кожен має два значення 0 або 1

Перетворення у дискретних полях Галуа – це додавання, множення, віднімання та ділення

Якщо взяти поле $GF(2^8)$, то фактично для нього додавання буде працювати як XOR для кожного числа, а множення буде працювати як звичайне множення плюс взяття по модулю від незвідного полінома. В RIJNDAEL цей поліном є $x^8 + x^4 + x^3 + x + 1$

5 Детально описати алгоритм зворотнього перетворення (дешифрування)

Для дешифрування нам потрібно спочатку отримати ключі для всіх раундів через Key Expansion, або ж розширення ключа. Такий самий алгоритм застосовується щоб отримати ключі і для шифрування. Фактично це набір операцій з матрицями та таблиця з даними. Це не стосується напряму дешифрування, але варто згадати

Для дешифрування виконуються всі операції шифрування, але в оберненому порядку та в спеціальній формі

Тут важливо замітити, що діаграми з різних джерел по різному групують операції, але про це трошки згодом.

Базовими операціями є:

- AddRoundKey – XOR з ключем раунду

- InvShiftRows – Інвертована ShiftRow, яка переставляє елементи в рядках в початкове положення

- InvMixColumns – Інвертована MixColumns, яка переставляє стовбці в початкове положення через спеціальну матрицю

- InvSubBytes – Інвертована SubBytes, яка перетворює біти на початкові через таблицю

Першим ділом виконується останній блок шифрування, який не включає mix columns в оберненому порядку. Він включає: AddRoundKey, InvShiftRows та InvSubBytes. AddRoundKey тут використовується з останнім ключем по індексу

Далі виконується 9(для 128) раундів. Кожен раунд включає в себе: AddRoundKey, InvMixColumns, InvShiftRows та InvSubBytes. AddRoundKey використовує ключі в оберненому порядку

Останнім виконується AddRoundKey з початковим ключем

Також є групування на діаграмах трошки по іншому. Виконується AddRoundKey, після виконується N – 1 циклів: InvShiftRows, InvSubBytes, InvMixColums та AddRoundKey. А після виконується додатковий раунд без InvMixColumns. Ключі використовуються з кінця до початку

Фактично при обох підходах результат буде однаковий, відрізняється лише візуалізація

Діаграми показані на рисунках 1 та 2. Діаграму 1 взято з “AES IP for Hybrid Cryptosystem RSA-AES” Anane Nadjia, Anane Mohamed. Діаграму 2 взято з “Design and Implementation A different Architectures of mixcolumn in FPGA” Sliman Arrag , Abdellatif Hamdoun, Abderrahim Tragha and Salah eddine Khamlich

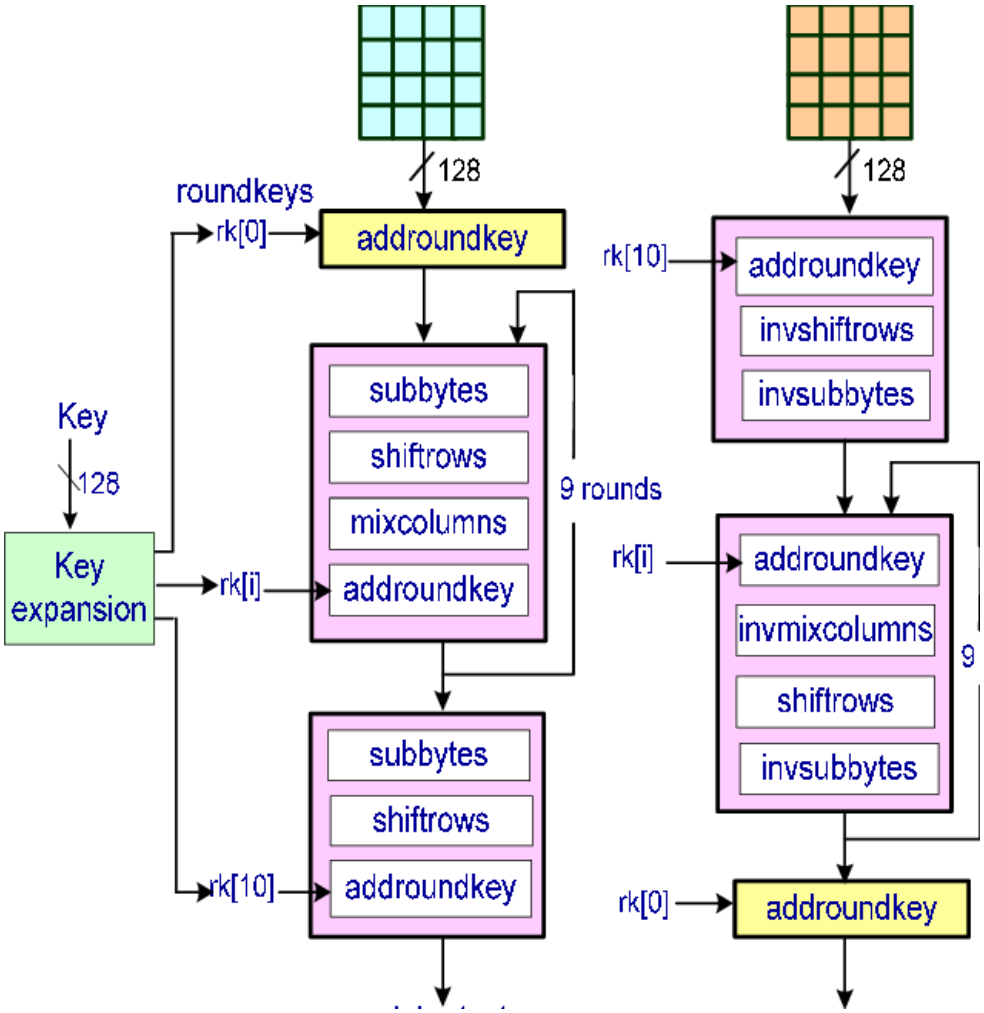


Рисунок 1 – Діаграма розшифрування 1

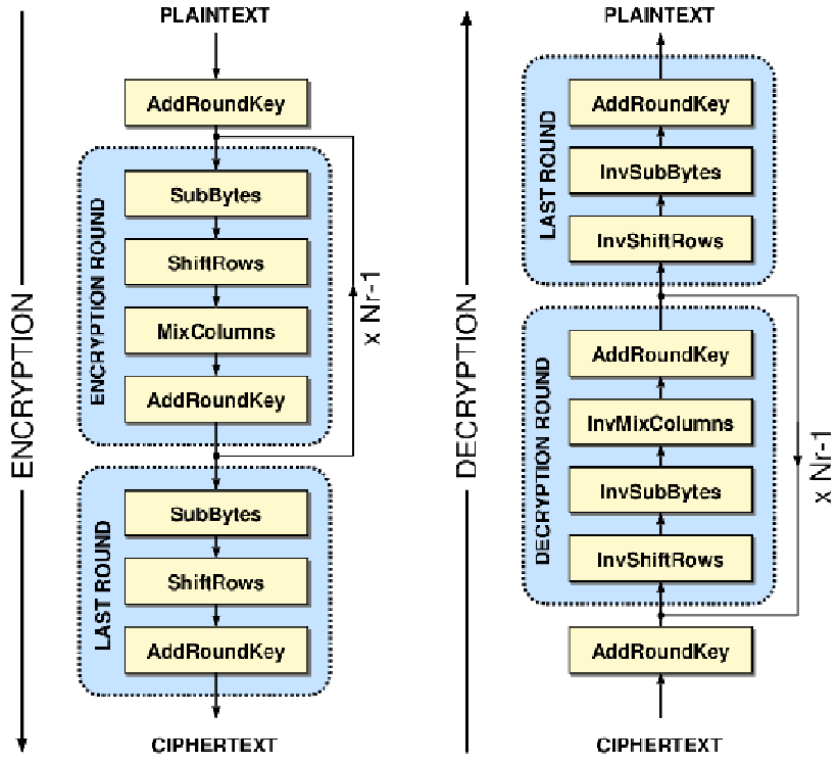


Рисунок 2 - Діаграма розшифрування 1

6 Зрозуміти та детально описати відмінності алгоритму AES-256

AES-128 використовує 128-бітний ключ та 10 раундів шифрування, а AES-256 використовує 256-бітний ключ та 14 раундів

В AES існує три стандартних розміри ключів: 128, 192 та 256 біт

Причина чому існує три ключі пов’язана з історією виникнення AES та тим, де він застосовується, і це включає в себе армію Сполучених Штатів Америки. До моменту коли з’явились ком’ютери більшість алгоритмів шифрування можна було зламати, а робити їх більш безпечними було дуже складно та повільно. Тому в армії на той момент вирішили використовувати три рівні захисту. Інформація яка повинна бути точно безпечна повинна шифруватись найбільш сильним алгоритмом, і тому складним, а щось не настільки важливе можна зашифрувати алгоритмом простіше.

Фактично, про нижній рівень припускалось, що він слабким в деякому моменті, але ця слабість не є обо’язковою

Коли створювали AES вони створили три рівні, бо так потрібно, але вони вирішили, що навіть найменший рівень буде неможливо зламати з передбачуваними технологіями. 128 біт є достатньо безпечним та його і так складно зламати. Враховуючи, що 256 на +-40% повільніше.

Цитата: “Therefore AES accepts 256-bit keys because of bureaucratic lassitude: it was easier to demand something slightly nonsensical (a key size overkill) than to amend military regulations.”

Посилання на джерело: https://security.stackexchange.com/questions/14068/why-most-people-use-256-bit-encryption-instead-of-128-bit

Що стосується технічних деталей – логіка алгоритму не змінюється, використовується все те саме, але блок і ключ мають довжину в 256 біт. Звичайно такий алгоритм буде повільнішим, оскільки у нас більше елементів

А також гарне пояснення чому 128 біт є достатньоз з поточними технологіями: https://security.stackexchange.com/questions/6141/amount-of-simple-operations-that-is-safely-out-of-reach-for-all-humanity/6149#6149

7 Керуючись довідковими матеріалами та мережею Інтернет, познайомитись із блочним симетричним шифром "Калина"(ДСТУ 7624:2014) та порівняти його з AES

Шифр “Калина” побудований на основі Substitution-Permutation Network, так само як і AES, а тому має дуже схожу структуру. Можна навіть сказати, що цей шифр є удосконаленням та розвитком AES

Вхідний блок може мати довжину в 128/256/512 бітів

Ключ може мати довжину в 128/256/512 бітів

Кількість раундів складає 10/14/18

Такі розміри блоку та ключа обумовлені тим, що цей шифр оптимізований до 64-бітного виконання

У цього шрифта є декілька ключових відмінностей від AES

Перша це замінена процедура SubBytes, або ж підстановка байтів, яка тепер використовує 4 таблиці заміни. Кожен рядок використовує свою таблицю по індексу, тобто 0 = 0, 1 = 1 …, 3 = 3, 4 = 0 і так далі. Як і з AES можна використовувати не стандартні таблиці, можна також додавати таблиці. З деяких джерел я чув, що таблиць може бути 8, але стандартно 4

Використовується інший алгоритм для зсуву рядків. Цей зсув залежить від довжини блоку. Таблиця зсуву показана на рисунку 3

Номер рядка	Значення зсуву, байтів		
	Довжина блоку 128 бітів	*Довжина блоку 256 бітів*	*Довжина блоку 512 бітів*
0	0	0	0
1	0	0	1
2	0	1	2
3	0	1	3
4	1	2	4
5	1	2	5
6	1	3	6
7	1	3	7

Рисунок 3 – Залежність зсуві від довжини блоку

Процедура MixColumns використовує інші матриці для множення, які побудовані на інших незвіданих поліномах

Використовується інша процедура розгортання ключа

Використовується множення за модулем 2 та множення за модулем 2^64

Схема роботи шифру показана на рисунку 4

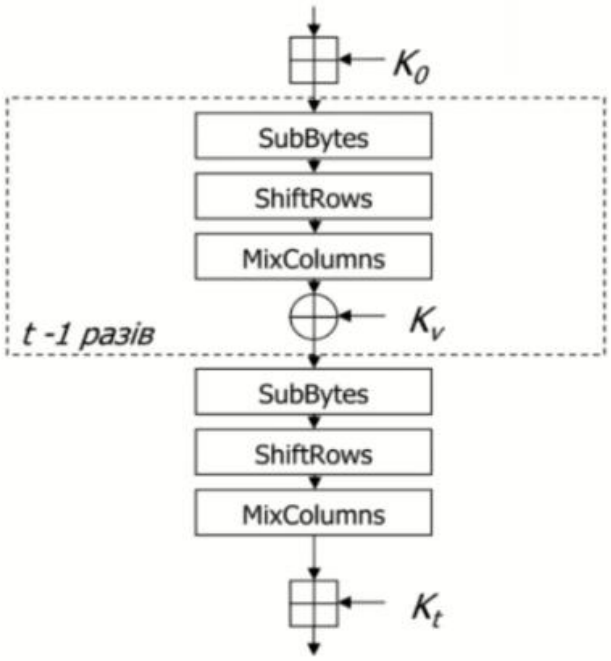


Рисунок 4 – Шифр Калина

Тобто, XOR використовується всередині раундів, але на початку шифрування та в самому останньому раунді використовується множення за модулем 2^64

Процедура розгортання ключів використовує ті самі блоки, які використовуються в самому алгоритмі. Схема розгортання ключів показана на рисунку 5

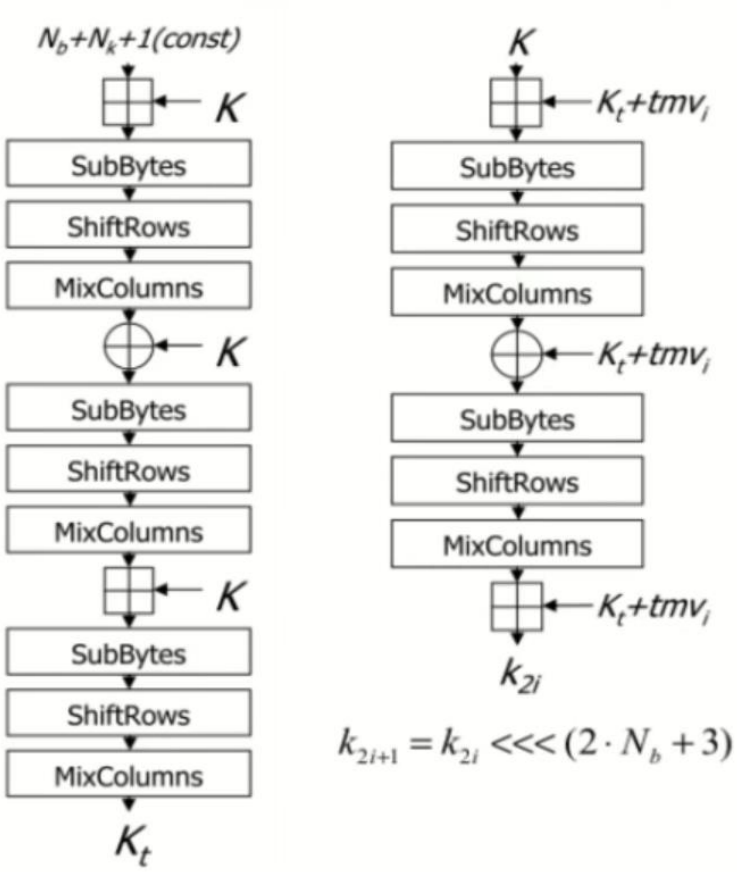
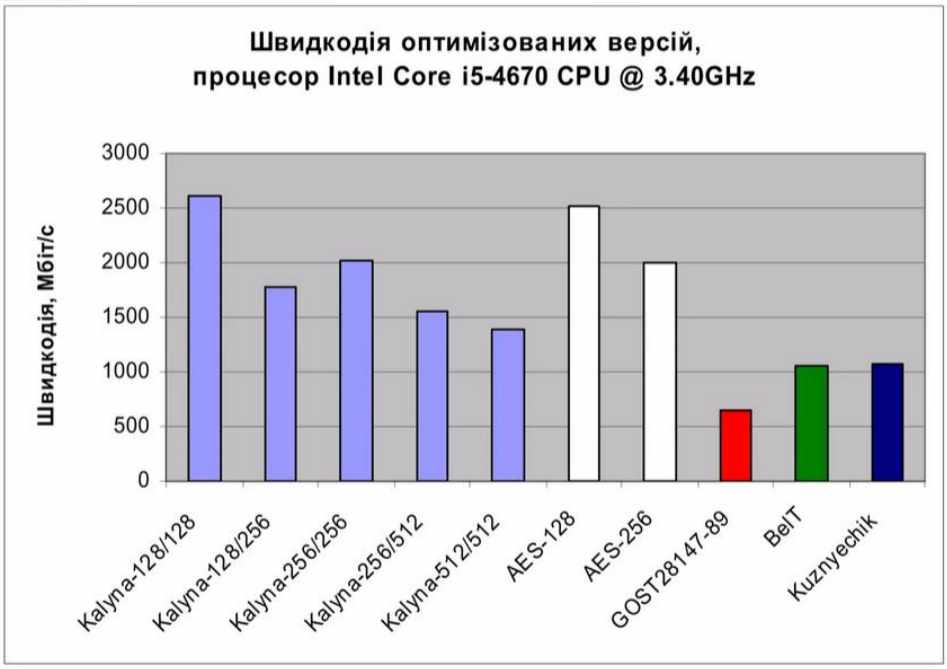


Рисунок 5 – Розгортання ключа в шифрі Калина

Порівняння швидкодії AES та Калина показані на рисунку 6



ОС: Linux 64 bit, компілятор gcc version 4.9.2 (Ubuntu 4.9.2-0ubuntu1~12.04, 30-Oct-2014)
https://github.com/Roman-Oliynykov/ciphers-speed/

Рисунок 6 – Порівняння швидкодії в шифрі Калина

Посилання: https://www.slideshare.net/slideshow/kalyna/48610310#15, https://learn.ztu.edu.ua/pluginfile.php/406458/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F7.pdf та https://youtu.be/Xhz6c7m7puU

8 Висновки

Я ознайомитись з алгоритмом симетричного блокового шифрування RIJNDAEL(AES) та Калина.