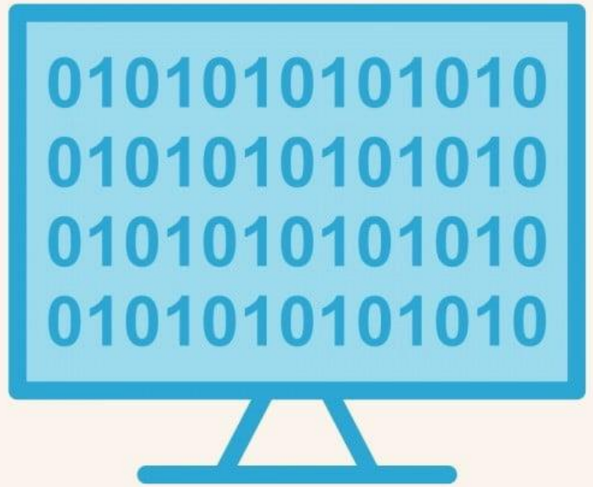
ЛЕКЦІЯ 7

ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Потокові симетричні шифри

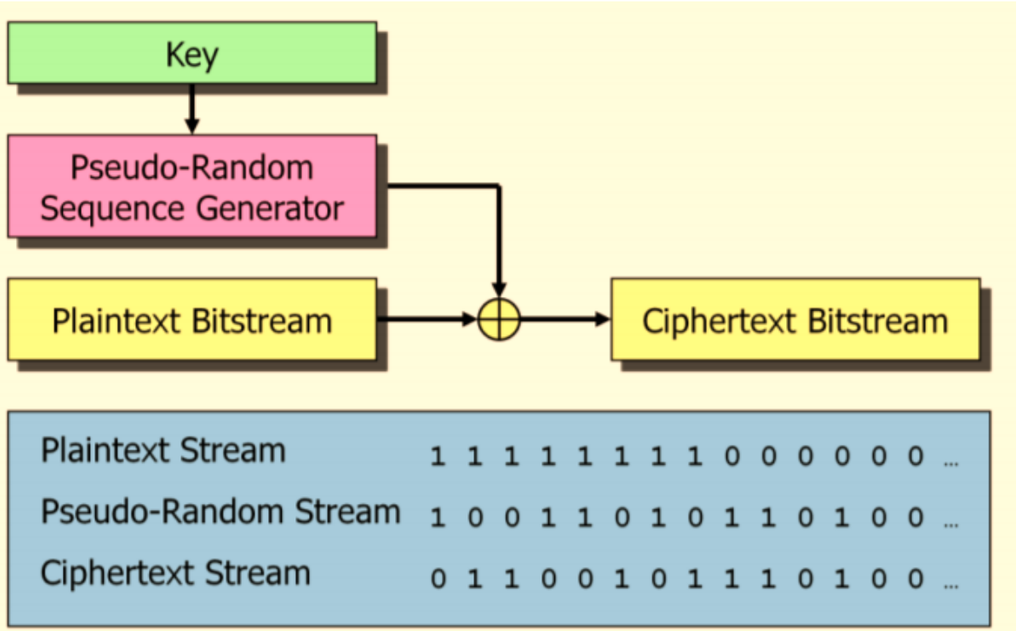


План

1. Загальні відомості про потокові шифри
2. Генератори псевдовипадкових чисел
3. Потоковий шифр RC4
4. Потоковий шифр ChaCha20
5. Інші потокові шифри

1. Загальні відомості про потокові шифри

Потоковий шифр – шифр, що перетворює кожен символ (літеру, біт або байт) відкритого тексту у символ шифротексту, залежно від ключа та розташування символів у тексті



Ключовий потік (гама) – це бітова послідовність, що визначається за допомогою ключа шифру

1. Загальні відомості про потокові шифри

Генератор ключів видає потік бітів K_i, які будуть використовуватися як гама. Джерело повідомлень генерує біти відкритих даних P_i, які додаються за модулем *2* з гамою, внаслідок чого виходять біти зашифрованих даних C_i:

$$C_i = P_i \oplus K_i$$

$$P_i = C_i \oplus K_i$$

1. Загальні відомості про потокові шифри

Синхронні потокові шифри

Синхронні потокові шифри – це шифри, у яких потік ключів генерується незалежно від відкритого й зашифрованого повідомлення

+ Відсутність ефекту поширення помилок (якщо тільки один біт спотворений, то тільки він буде розшифрований неправильно)

+ Захист від вставок і вилучень фрагментів зашифрованого повідомлення, оскільки вони призведуть до втрати синхронізації й будуть виявлені

– Можлива підміна окремих бітів зашифрованого повідомлення

1. Загальні відомості про потокові шифри

Потокові шифри, що самосинхронізуються

Потокові шифри, що самосинхронізуються (асинхронні потокові шифри) – це шифри, в яких потік ключів створюється функцією ключа й фіксованою кількістю знаків зашифрованого повідомлення (використовується *n* попередніх знаків шифротексту для обчислення потоку ключа)

\+ Дешифрувальний генератор, прийнявши *n* бітів, автоматично синхронізується із шифрувальним генератором

\+ Розсіювання статистики відкритого тексту

– Чутливий до розкриття повторною передачею

2. Генератори псевдовипадкових чисел

Стійкість потокових шифрів цілком залежить від якості й криптографічної стійкості генератора псевдовипадкових чисел, за допомогою якого отримується потік ключа

Генератор псевдовипадкових чисел (ГПВЧ) – це пристрій або алгоритм, який за заданими параметрами генерує послідовність *псевдовипадкових чисел* (ПВЧ)

2. Генератори псевдовипадкових чисел

Послідовність випадкових чисел – послідовність, сформована за допомогою фізичного процесу, результат якого є непередбачуваний і не може бути повторно відтворений

Послідовність псевдовипадкових чисел – відтворювана послідовність, сформована за допомогою детермінованого алгоритму (незалежно від способу його реалізації), що володіє статистичними властивостями послідовностей випадкових чисел

2. Генератори псевдовипадкових чисел

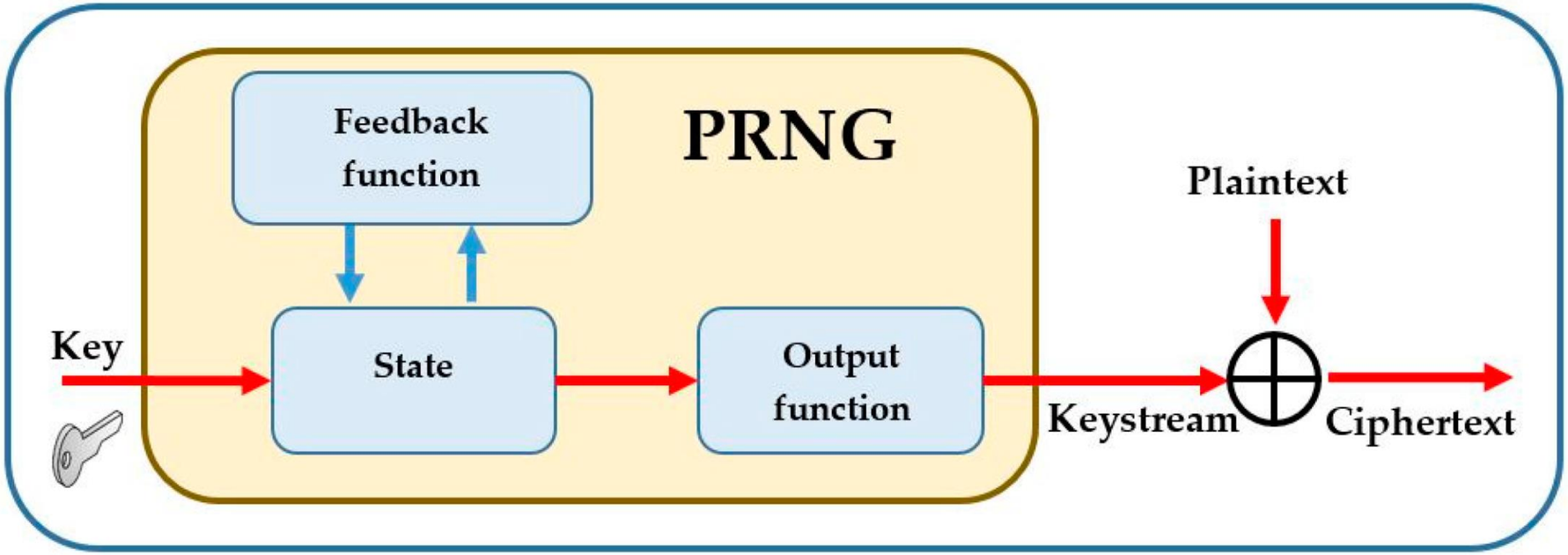
Чи можна вважати дану послідовність псевдовипадковою? Які будуть наступні елементи?

12, 14, 14, 17, 16, 20,…

2. Генератори псевдовипадкових чисел

Метою використання ГПВЧ у потокових шифрах є отримання нескінченної ключової послідовності, за використання відносно малої довжини самого початкового ключа.

Ключ, який породжує гаму також називають зародком (seed)



2. Генератори псевдовипадкових чисел

Найважливішою характеристикою ГПВЧ є довжина періоду повторення, після якого випадкові числа, на виході ГПВЧ почнуть повторюватися

Псевдовипадкову послідовність можна використовувати як ключ тільки до повтору, інакше потоковий шифр виродиться в нестійкий алгоритм звичайного XOR

2. Генератори псевдовипадкових чисел

Вимоги до ГПВЧ

1) Непередбачуваність (неможливо визначити наступне число з імовірністю, більшою за $\frac{1}{m}$, де m – потужність алфавіту генератора);

2) Відтворюваність дозволяє повторити (за відомих початкових значень) з абсолютною точністю послідовність на виході ГПВЧ в довільний момент часу та довільну кількість разів;

3) Великий період повторення (настільки великий, що його неможливо відтворити сучасними технічними засобами);

4) Числа, що генеруються мають бути статистично рівномірно розподілені.

2. Генератори псевдовипадкових чисел

Лінійний конгруентний генератор (LCG)

Для обчислення чергового числа x_{i+1} використовується формула:

$$x_{i+1} = (a \cdot x_i + b) mod\ m,$$

де a, b, m – деякі константи, а x_i – попереднє псевдовипадкове число, а x_0 – початкове значення (seed)

Якщо параметри LCG обрані правильно, то генератор буде породжувати випадкові числа з максимальним періодом, що дорівнює m

2. Генератори псевдовипадкових чисел

Приклад 2.1: $a = 5, b = 3, m = 11, x_0 = 1.$

$x_1 = (5 \cdot 1 + 3) mod\ 11 = 8;$
$x_2 = (5 \cdot 8 + 3) mod\ 11 = 10;$
$x_3 = (5 \cdot 10 + 3) mod\ 11 = 9;$
$x_4 = (5 \cdot 9 + 3) mod\ 11 = 4;$
$x_5 = (5 \cdot 4 + 3) mod\ 11 = 1;$
$x_6 = (5 \cdot 1 + 3) mod\ 11 = 8;$
$x_7 = (5 \cdot 8 + 3) mod\ 11 = 10;$
$x_8 = (5 \cdot 10 + 3) mod\ 11 = 9.$

Лінійні конгруентні генератори не рекомендують використовувати, оскільки криптоаналітики навчилися відновлювати всю послідовність ПВЧ із кількох значень

2. Генератори псевдовипадкових чисел

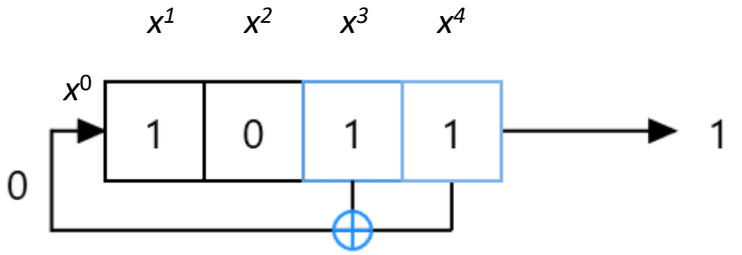
ГПВЧ на основі регістрів зсуву з лінійним зворотним зв'язком (LFSR)

Зворотний зв’язок є функцією XOR певних бітів регістра, які називають ***відводами*** (taps)

Розташування відводів для зворотного зв'язку може бути представлене многочленом з коефіцієнтами 0 або 1

Наприклад, для 4-бітового LFSR многочлен зворотного зв'язку:

$$x^4 + x^3 + 1$$



2. Генератори псевдовипадкових чисел

Алгоритм LFSR

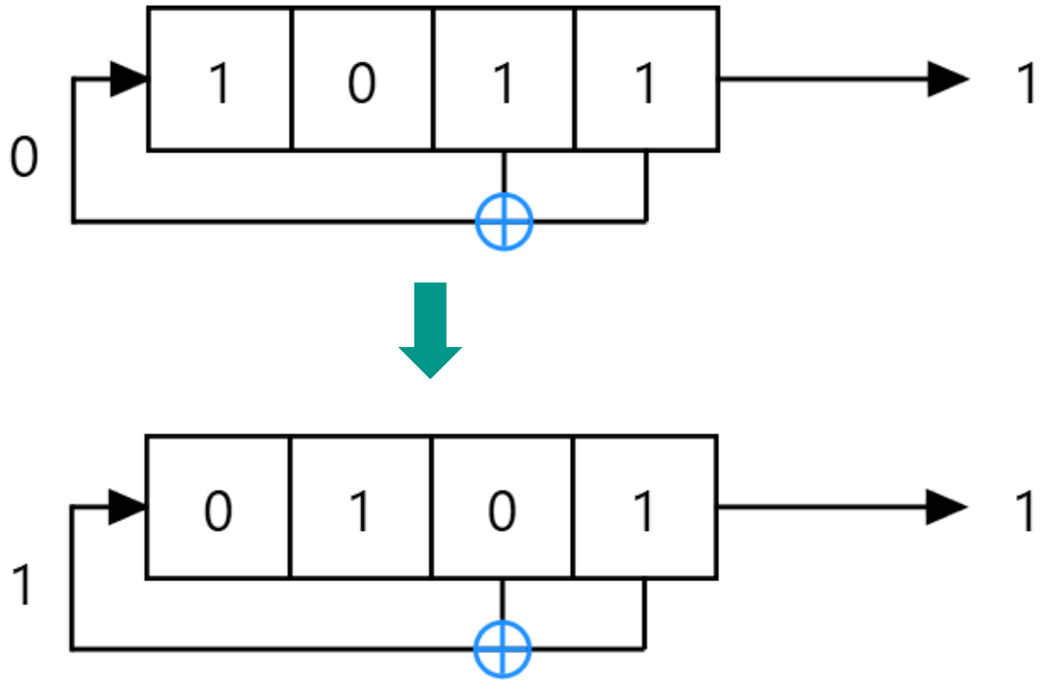
1. У регістр записується початкове значення.
2. Обчислюється XOR відповідних бітів (відводів).
3. Обчислений біт записується до першої позиції регістру ліворуч.
4. Усі решта біт зсуваються на одну позицію праворуч.
5. Останній біт з правого боку усувається з регістру і стає черговим бітом псевдовипадкової послідовності.
6. Алгоритм повторюється з 2 кроку.

2. Генератори псевдовипадкових чисел

Приклад 2.2:

Номер стану	Внутрішній стан Q R S T	Результат обчислення S⊕T	Біт, псевдовипадкової послідовності
1	1 0 1 1	0	1
2	0 1 0 1	1	1
3	1 0 1 0	1	0
4	1 1 0 1	1	1
5	1 1 1 0	1	0
6	1 1 1 1	0	1
7	0 1 1 1	0	1
8	0 0 1 1	0	1
9	0 0 0 1	1	1
10	1 0 0 0	0	0

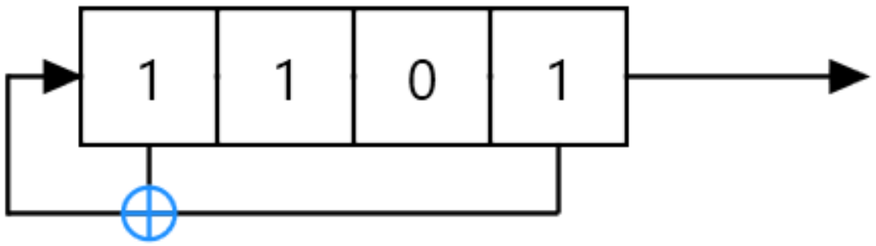
2. Генератори псевдовипадкових чисел



LFSR розміром *n* бітів може перебувати в одному з $2^n–1$ станів. Тому, теоретично можна генерувати псевдовипадкову послідовність із максимальним періодом $2^n–1$.

2. Генератори псевдовипадкових чисел

Яким буде ключовий потік, що генерує LFSR, якщо його многочлен зворотного зв'язку $x^4 + x + 1$, а початковий стан 1101?



а) 1101010110;

б) 1101011001;

в) 1011001000;

г) 0010101011.

2. Генератори псевдовипадкових чисел

ГПВЧ на основі алгоритму BBS

Широке поширення в криптографії набув алгоритм Блюма – Блюма – Шуба (від прізвищ авторів – *L. Blum, M. Blum, M. Shub*), який називають ще генератором квадратичних лишків.

Послідовність, сформована за допомогою цього методу, має статистичні властивості, близькі до генераторів випадкових чисел, а метод є досить крипостійким

2. Генератори псевдовипадкових чисел

Алгоритм BBS

1. Обирають два великих простих числа p та q, які конгруентні. При діленні цих чисел на *4* повинен виходити однаковий залишок *3*.

2. Обчислюється $M = p \cdot q$ – *ціле число Блюма*.

3. Вибирається інше випадкове ціле число x, взаємно просте з M.

4. Обчислюється $x_0 = x^2 mod\ M$ – *стартове число генератора*.

5. На кожному *n*-му кроці обчислюється $x_{n+1} = x_n^2 mod\ M$.

6. Результатом *n*-го кроку є один (зазвичай молодший) біт числа x_{n+1}.

2. Генератори псевдовипадкових чисел

Приклад 2.3: Нехай $p = 11$, $q = 19$, (переконуємося, що $11\ mod\ 4 = 3$, $19\ mod\ 4 = 3$).

Тоді $M = p \cdot q = 11 \cdot 19 = 209$.

Виберемо x, взаємно просте з M: нехай $x = 3$.

Обчислимо стартове число генератора x_0:

$$x_0 = x^2 mod\ M = 3^2 mod\ 209 = 9\ mod\ 209 = 9$$

Далі обчислимо перші десять чисел x_i за алгоритмом *BBS*. Як випадковий біт будемо брати молодший біт у двійковому записі числа x_i.

2. Генератори псевдовипадкових чисел

$x_1 = 9^2 \bmod 209 = 81 \bmod 209 = 81$ – молодший біт: *1*;

$x_2 = 81^2 \bmod 209 = 6561 \bmod 209 = 82$ – молодший біт: *0*;

$x_3 = 82^2 \bmod 209 = 6724 \bmod 209 = 36$ – молодший біт: *0*;

$x_4 = 36^2 \bmod 209 = 1296 \bmod 209 = 42$ – молодший біт: *0*;

$x_5 = 42^2 \bmod 209 = 1764 \bmod 209 = 92$ – молодший біт: *0;*

$x_6 = 92^2 \bmod 209 = 8464 \bmod 209 = 104$ – молодший біт: *0;*

$x_7 = 104^2 \bmod 209 = 10816 \bmod 209 = 157$ – молодший біт: *1*;

$x_8 = 157^2 \bmod 209 = 24649 \bmod 209 = 196$ – молодший біт: *0*;

$x_9 = 196^2 \bmod 209 = 38416 \bmod 209 = 169$ – молодший біт: *1*;

$x_{10} = 169^2 \bmod 209 = 28561 \bmod 209 = 137$ – молодший біт: *1*.

3. Потоковий шифр RC4

RC4 (Rivest Cipher 4, Ron's Code) – потоковий шифр, який розробив Рональд Рівест у 1987 р.

Шифр RC4 застосовувався в деяких широко поширених стандартах і протоколах таких, як WEP, WPA і TLS (до 2015 року)

Рональд Лінн Рівест

3. Потоковий шифр RC4

Ініціалізація матриці стану і масиву ключів

Матриця стану (256 байт) початково заповнюється значеннями:

$S[0] = 0, S[1] = 1, S[2] = 2, \ldots, S[254] = 254, S[255] = 255.$

Ключем є послідовність байтів, що записується у тимчасовий масив *T*:

$T[0], T[1], T[2], \ldots, T[254], T[255].$

Якщо ключ шифрування має точно *256* байтів, то байти копіюються в масив *T*, інакше байти повторюються, поки не заповниться масив *T*

3. Потоковий шифр RC4

Перестановка матриці стану на основі значень ключа

Кожен черговий елемент S_i обмінюється місцями з елементом S_j, номер j якого визначається сумою номера елементу j, самого елементу S_i та елементу ключа T_i.

Значення лічильників i та j спочатку дорівнюють 0.

```
for
    i = 0 to 255 do S[i] = i;
T[i] = K[i mod k - len];
```

```
j = 0;
for
    i = 0 to 255 do
    {
        j = (j + S[i] + T[i])mod 256;
        Swap(S[i], S[j]);
    }
```

3. Потоковий шифр RC4

Перестановка матриці стану і генерація ключового потоку

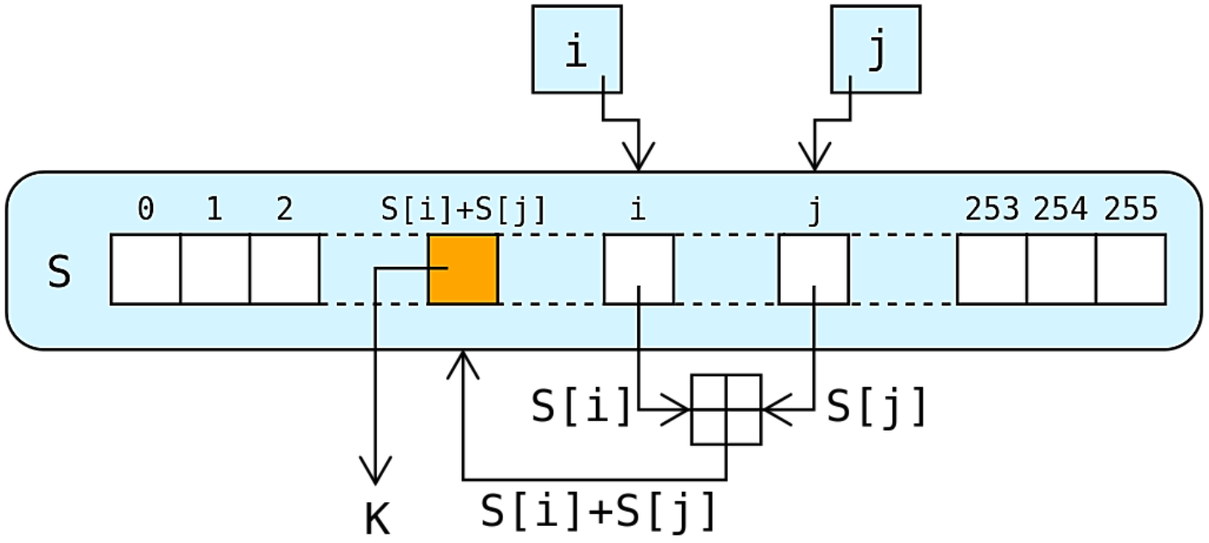
Два елементи матриці стану міняються місцями на основі двох індивідуальних змінних *i* та *j*, що обчислюються за відповідними формулами.

Потім сума значень елементів S_i та S_j визначає індекс *t* елементу S_t, який використовуватиметься як ключ *k*.

```
i, j = 0;
while (true)
     i = (i + 1)mod 256;
j = (j + S[i])mod 256;
Swap(S[i], S[j]);
t = (S[i] + S[j])mod 256;
k = S[t];
```

3. Потоковий шифр RC4

Генерація ключового потоку (ПВЧ) у RC4



Шифрування RC4

При шифруванні байти *K* додають за модулем 2 з байтами відкритого тексту

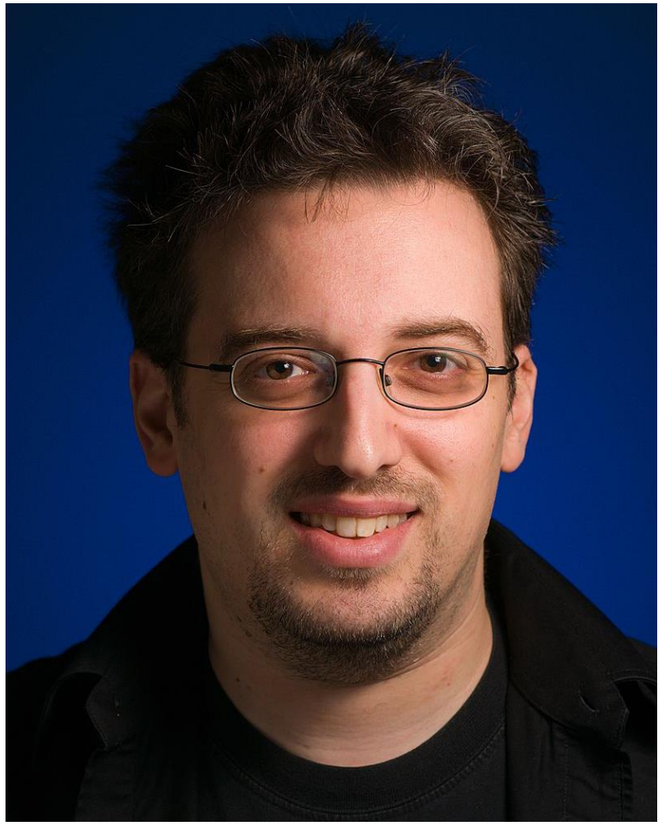
Приклад роботи RC4

4. Потоковий шифр ChaCha20

ChaCha20 – потоковий шифр, розроблений Даніелем Бернштейном (Daniel Bernstein) у 2008 р.

ChaCha20 (разом із алгоритмом автентифікації Poly1305) широко застосовується в TLS-з’єднаннях як альтернатива AES.

ChaCha20 втричі швидше за AES



Даніель Бенштейн

4. Потоковий шифр ChaCha20

Матриця стану (**512** біт) має 4 рядки та 4 стовпці, кожна комірка містить **32** біти (4 байти)

C (**128** біт) – константи 61707865, 3320646E,79622D32, 6B206574

K (**256** біт) – ключ

CTR (**64** біти) – лічильник блоку

IV (**64** біти) – вектор ініціалізації

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

Початкові дані записуються у матрицю стану у форматі little endian

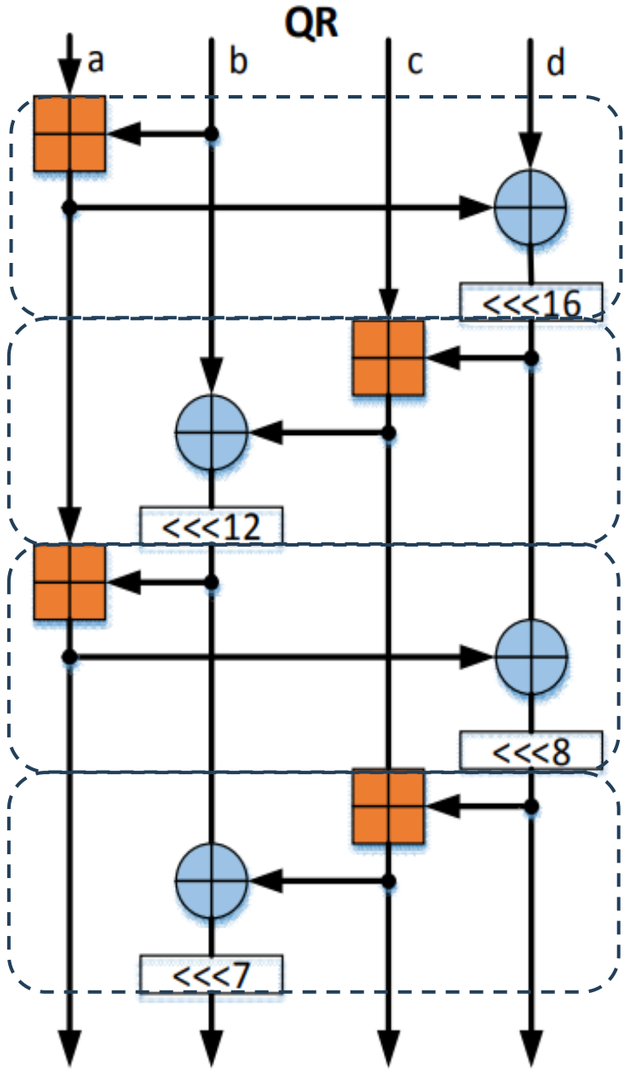
4. Потоковий шифр ChaCha20

Функція QuaterRound – QR(a, b, c, d)

Над чотирма 32-бітними числами a, b, c та d виконуються такі операції:

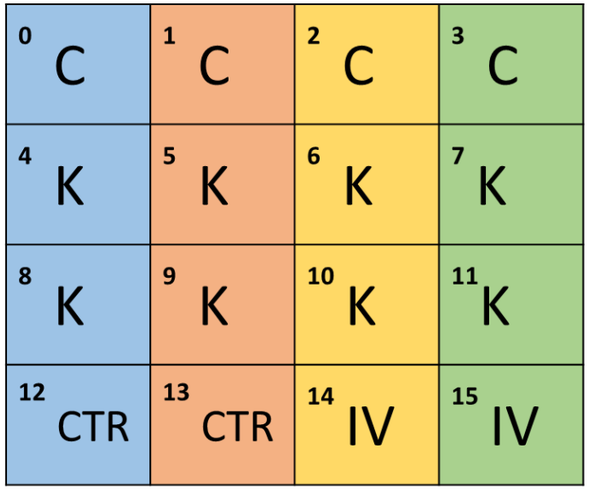
- арифметичне додавання за модулем 2^{32};
- циклічний зсув вліво на задане число позицій u (<<< u);
- побітове додавання за модулем 2 (XOR)

QR змінює чотири значення стану a, b, c, d. Після кожного раунду увесь стан зміниться

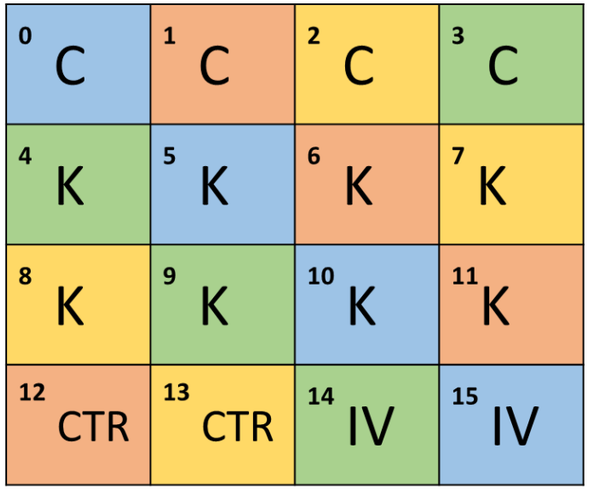


4. Потоковий шифр ChaCha20

На кожному із **20** раундів **4** рази викликається функція QR та виконуються операції, які обробляють матрицю стану за стовпцями (непарні раунди) і діагоналями (парні раунди)



1. QR(0, 4, 8, 12)
2. QR(1, 5, 9, 13)
3. QR(2, 6, 10, 14)
4. QR(3, 7, 11, 15)

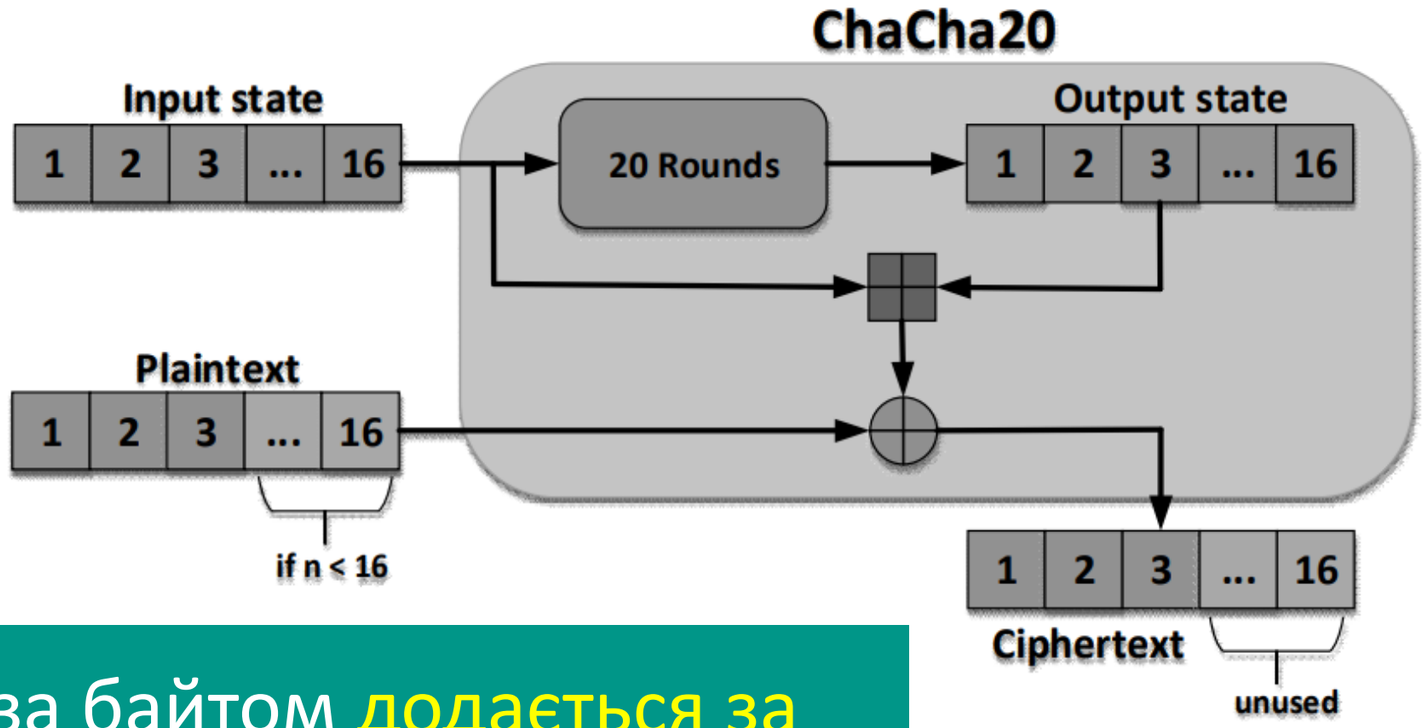


5. QR(0, 5, 10, 15)
6. QR(1, 6, 11, 12)
7. QR(2, 7, 8, 13)
8. QR(3, 4, 9, 14)

4. Потоковий шифр ChaCha20

Шифрування ChaCha20

Після 20 раунду до отриманого результату арифметично додають початкове значення матриці стану



Потім матриця байт за байтом додається за модулем 2 (XOR) до відкритого тексту

4. Потоковий шифр ChaCha20

Шифрування ChaCha20

Якщо розмір відкритого тексту більший за 512 біт, то потрібно послідовно викликати функцію QR із тим самим ключем і вектором ініціалізації, але кожен раз збільшуючи значення лічильника блоку CTR на 1

Дешифрування повністю аналогічне шифруванню

5. Інші потокові шифри

Потоковий шифр Salsa20

Програмно-орієнтований алгоритм Salsa20 став переможцем міжнародного конкурсу eSTREAM. Для ініціалізації внутрішнього стану використовується ключ довжиною 256 біт, 64-бітний nonce та 64-бітний лічильник блоку. Максимальна довжина псевдовипадкової ключової послідовності дорівнює 2^{70} біт

5. Інші потокові шифри

Потоковий шифр SNOW 2.0

Потоковий симетричний шифр SNOW 2.0 є генератором ключових потоків, який використовує як вхідні дані 128 або 256-бітовий секретний ключ K і 128-бітовий вектор ініціалізації IV. На сьогодні цей шифр є стандартизованим та являє собою один з найбільш швидких програмно орієнтованих потокових шифрів

5. Інші потокові шифри

Потоковий шифр Trivium

Trivium – це симетричний апаратно-орієнтований паралельний потоковий шифр. Trivium найбільш простий шифр проекту eSTREAM, який демонструє відмінні результати криптостійкості. Trivium призначений для генерації 264 біт ключового потоку з 80 біт секретного ключа і 80 біт вектору ініціалізації. Шифр є біт-орієнтованим

5. Інші потокові шифри

Потоковий шифр Grain

Grain – синхронний потоковий апаратно-орієнтований шифр. Залежно від апаратної реалізації може бути біт-орієнтованим або слово-орієнтованим. На вхід подається ключ довжиною 80 біт та IV довжиною 64 біти. Довжина ключового потоку, який може бути вироблений на одній парі ключ/вектор – 2^{44} біт