УДК 004.056.55

О.О. КУЗНЕЦОВ, д-р техн. наук, І.Д. ГОРБЕНКО, д-р техн. наук,
Ю.І. ГОРБЕНКО, канд. техн. наук, А.М. ОЛЕКСІЙЧУК, д-р техн. наук, В.А. ТИМЧЕНКО

МАТЕМАТИЧНА СТРУКТУРА ПОТОКОВОГО ШИФРУ *СТРУМОК*

Вступ

Важливим механізмом криптографічного захисту інформації є потокове симетричне шифрування [1, 2]. Воно застосовується для забезпечення послуги конфіденційності та цілісності (як додаткової послуги) інформації під час обробки інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах [1].

Останніми роками вимоги до сучасних алгоритмів потокового шифрування істотно зросли [3 – 5]: з одного боку, потрібно забезпечувати високу швидкість криптографічного перетворення (понад 10 Гбит/с), з іншого – необхідно ефективно протистояти новітнім методам криптографічного аналізу, в тому числі із застосуванням квантових методів обчислення.

Отже розробка, дослідження та поступове впровадження нових методів потокового симетричного шифрування є актуальною та надзвичайно важливою науково-прикладною проблемою національного рівня.

Зазвичай операція потокового шифрування є побітовою операцією XOR між ключовим потоком і повідомленням. У ISO/IEC 18033-4:2011 описано вихідні функції для різних потокових шифрів, та певні генератори псевдовипадкових чисел, які призначено для захисту інформації з обмеженим доступом, зокрема, для забезпечення конфіденційності інформації під час її обробки [6]. Метою цієї роботи є викладення основних результатів з розробки нового генератору псевдовипадкових чисел (ключового потоку), який позначено «Струмок», та який пропонується у якості кандидата на національний стандарт симетричного шифрування в Україні [7 – 18]. Генератор «Струмок» забезпечує високу швидкість формування ключового потоку (понад 10 Гбіт/с), яка перевищує більшість відомих алгоритмів, та придатний до застосування в постквантовому середовищі [7 – 9].

Загальні параметри потокового шифру

В основі алгоритму *Струмок* лежить класична схема підсумовуючого генератора [1, 2, 6, 7], подібна генератору *SNOW-2.0*, який визначено в ISO/IEC 18033-4:2011 [6]. Алгоритм *Струмок* використовує 256-бітний вектор ініціалізації *IV* та 256-бітний або 512-бітний секретний ключ *K* і забезпечує високий та надвисокий рівень стійкості із врахуванням можливого застосування квантового криптографічного аналізу. Криптоалгоритм орієнтований на 64-розрядні обчислювальні системи, отже розмір слова визначено рівним 64 бітам.

Основними структурними компонентами генератору є регістр зсуву з лінійним зворотнім зв'язком (linear feedback shift register, *LFSR*) та скінченний автомат (finite-state machine, *FSM*), в якому виконується нелінійне перетворення. Вхідні дані використовуються для ініціалізації змінної стану $S_i (i \geq 0)$, яка складається з вісімнадцяти 64-бітових блоків, до складу яких входить дві компоненти:

– 16 змінних $s^{(i)}$ – комірок регістра зсуву з лінійним зворотнім зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, ..., s_0^{(i)})$;

– два регістри скінченного автомату $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$.

На виході отримуємо ключовий потік (гаму шифру), який формується з 64-бітових слів Z_i. Схематичне зображення функціонування генератора ключових потоків *Струмок* в довільний момент часу *i* наведено на рис. 1. Змінну часової залежності *i* не наведено.

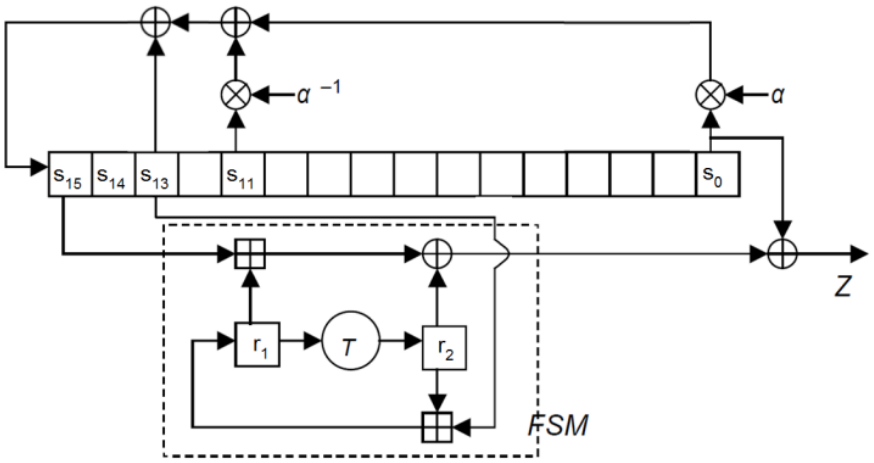

Рис. 1. Схематичне зображення генератору ключових потоків *Струмок* у режимі генерації гами шифру (ключового потоку)

Відводи зворотного зв'язку у LFSR будується за примітивним над полем $GF(2^{64})$ поліномом

$$f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha,$$

де α є коренем примітивного над полем $GF(2^8)$ поліному

$$g(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2 z^5 + \beta^{224}z^4 + \beta^{70}z^3 + \beta^2.$$

В свою чергу поле $GF(2^8)$ будується за примітивним над полем $GF(2)$ поліномом

$$p(y) = y^8 + y^4 + y^3 + y^2 + 1,$$

а коефіцієнти поліному $g(z)$ подаються через ступінь примітивного елементу β поля $GF(2^8)$, тобто β – корінь поліному $p(y)$.

Таким чином, маємо вежу полів:

$$GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024}),$$

де

– поле $GF(2^{1024})$ задається відводами зворотного зв'язку LFSR як факторкільце $GF(2^{64})[x]/(f(x))$,

– поле $GF(2^{64})$ задається як факторкільце $GF(2^8)[z]/(g(z))$,

– поле $GF(2^8)$ задається як факторкільце $GF(2)[y]/(p(y))$.

Отже період вихідної послідовності *LFSR* є максимальним і дорівнює $2^{1024} - 1$.

Структурно в алгоритмі *Струмок* можна виділити три основні функції:

– функція ініціалізації *Init*, яка приймає в якості вхідних даних ключ *K* (256 біт або 512 біта) і вектор ініціалізації *IV* (256 біт), і виробляє початкове значення змінної стану

$$S_0 = (s^{(0)}, r^{(0)});$$

– функція наступного стану *Next*, яка приймає на вхід змінну стану

$$S_i = (s^{(i)}, r^{(i)})$$

і виробляє наступне значення змінної стану

$$S_{i+1} = (s^{(i+1)}, r^{(i+1)}).$$

Функція *Next* може виконуватися в двох режимах, в залежності від способу виконання ітерації – як частини реалізації або як частини нормального режиму генерації вихідних даних;

– функція ключового потоку *Strm*, що приймає на вході змінну стану $S_i = (s^{(i)}, r^{(i)})$ і виробляє на виході 64-бітний ключовий потік Z_i.

Функція ініціалізації внутрішнього стану *Init*

Функція ініціалізації внутрішнього стану *Init* описується наступним чином.

Вхід: 256 або 512-бітний ключ *K*, 256-бітний вектор ініціалізації *IV*.

Вихід: початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$.

Ключ для версії потокового шифру *Струмок-256* можна представити у вигляді чотирьох 64-бітних слів

$$K = \left(K_3, K_2, K_1, K_0\right)$$

а для 512-бітного ключа – у вигляді восьми 64-бітних слів

$$K = \left(K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0\right),$$

де K_3 та K_7, відповідно для 256 и 512 біт, найбільш значущі слова, а K_0 – найменш значущі.

Вектор ініціалізації можна представити у вигляді чотирьох 64-бітних слів

$$IV = \left(IV_3, IV_2, IV_1, IV_0\right),$$

де IV_3 – найбільш значуще слово, а IV_0 – найменш значуще.

1. В 16 комірок LFSR заноситься значення ключа.

Для версії з 256-бітним ключем виконуються операції:

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

Для версії з 512-бітним ключем *K* виконуються операції:

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

2. Виконуються 32 ініціюючих такти без генерації ключового потоку, тобто чотири повних циклів. Формально це подається наступним чином:

$$S_{-1} = Next^{32}(S_{-33}, INIT),$$

що означає 32 ітерації з виконання функції *Next* у режимі ініціалізації *INIT*, $S_{-33} = (s^{(-33)}, r^{(-33)})$ – обраховані на попередньому кроці значення змінної стану.

3. Розраховується початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$ за правилом: $S_0 = Next(S_{-1})$, тобто шляхом виконання функції *Next* у звичайному режимі.

4. Виводиться вихідне значення $S_0 = (s^{(0)}, r^{(0)})$.

Функція наступного стану *Next*

Функція стану *Next* описується наступним чином.

Вхід: Змінна стану $S_i = (s^{(i)}, r^{(i)})$, обраний режим (звичайний, або режим ініціалізації).

Вихід: Наступне значення змінної стану $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$.

1. Виконується нелінійна підстановка для оновлення значення регістру $r_2^{(i+1)}$ скінченного автомату. Для цього розраховується значення функції T: $r_2^{(i+1)} = T(r_1^{(i)})$.

2. Оновлюється значення регістру $r_1^{(i+1)}$ скінченного автомату. Для цього розраховується значення

$$r_1^{(i+1)} = r_2^{(i+1)} +_{64} s_{13}^{(i)},$$

де $+_{64}$ позначає операцію додавання цілих чисел за модулем 2^{64} (у схемі шифру на рис. 1 цю операцію позначено як ⊞).

3. Оновлюється значення 15 комірок *LFSR*

$$s_j^{(i+1)} = s_{j+1}^{(i)}$$

для всіх $j = 0, 1, \ldots, 14$.

4. Оновлюється значення 16-ї комірки *LFSR*. Якщо встановлено звичайний режим функції *Next*, значення цієї комірки обчислюється за правилом

$$s_{15}^{(i+1)} = (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)}.$$

Якщо встановлено режим ініціалізації *INIT* функції *Next*, значення обчислюється за правилом

$$s_{15}^{(i+1)} = FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \oplus (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)}.$$

Операції множення $\otimes$ на α та на α^{-1} та сутність функції FSM пояснюються далі.

5. Обчислюється та виводиться значення змінної стану $S_i = (s^{(i)}, r^{(i)})$.

Схематичне зображення генератору ключових потоків *Струмок* при виконанні функції *Next* у режимі ініціалізації представлено на рис. 2.

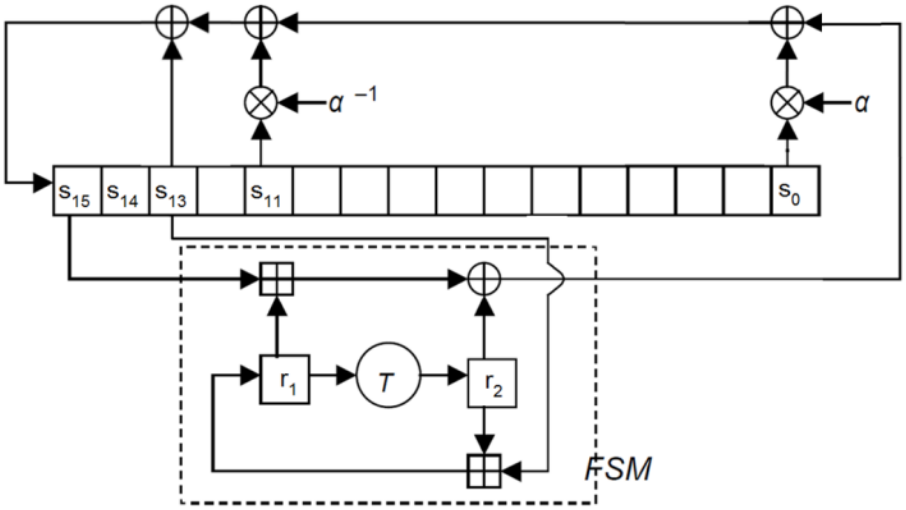

Рис. 2. Схематичне зображення генератору ключових потоків Струмок у режимі ініціалізації функції *Next*

Функція ключового потоку *Strm*

Функція ключового потоку *Strm* описується наступним чином.

Вхід: Змінна стану $S_i = (s^{(i)}, r^{(i)})$.

Вихід: 64-бітовий ключовий потік Z_i.

1. Обчислюється значення

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

2. Виводиться вихідне значення Z_i.

Функція скінченного автомату *FSM*

Функція скінченного автомату позначається як *FSM* (*x*, *y*, *z*) та описується наступним чином.

Вхід: три 64-бітових рядка *x*, *y* і *z*.

Вихід: 64-бітовий рядок *q*.

1. Обчислюється значення $q = (x +_{64} y) \oplus z$.

2. Виводиться вихідне значення *q*.

Функція нелінійної підстановки *T*

Функція нелінійної підстановки T реалізує перестановку елементів скінченного поля $GF(2^{64})$ за допомогою компонентів національного стандарту блокового симетричного криптоперетворення ДСТУ 7624:2014 [19].

Вхід: 64-бітовий рядок w.

Вихід: 64-бітовий рядок $T = T(w)$.

1. Вхідний 64-бітовий рядок w розбивається на підблоки w_j по 8 біт:

$$w = (w_7, w_6, w_5, w_4, w_3, w_2, w_1, w_0),$$

2. Для кожного підблоку w_j виконується підстановка з алгоритму ДСТУ 7624:2014 за допомогою чотирьох табличних перетворень $\pi_0, \pi_1, \pi_2, \pi_3$ (див. додаток Б). Виконання функції T за допомогою цих перетворень схематично (на прикладі підблоків w_j, що подано у шіснадцятковому вигляді) зображено на рис. 3.

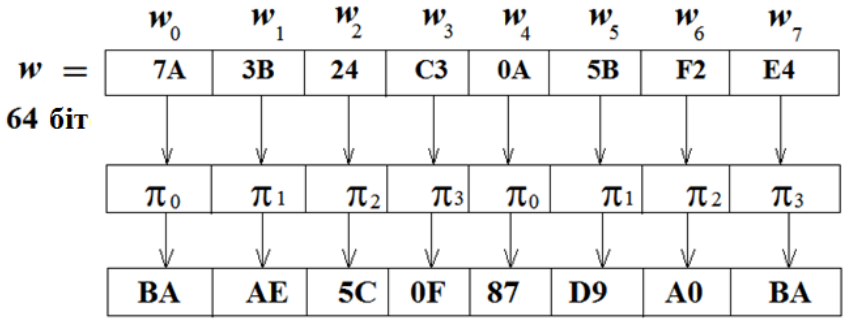

Рис. 3. Схематичне зображення виконання процедури підстановки $T = T(w)$

У результаті формується вихідний вектор $r = (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$:

$$r_j = \pi_{j \bmod 4}\left[w_j\right],$$

де $j = 0, 1, \ldots, 7$.

3. Обчислюється вектор

$$q = (q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0)$$

за правилом

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

де елементи матриці (подано у шістнадцятковому вигляді) та векторів r і q інтерпретуються як елементи скінченного поля $GF(2^8)$, яке задане як факторкільце $GF(2)[y]/(p(y))$.

Цю операцію можна записати у скороченому вигляді (як у ДСТУ 7624:2014):

$$q_i = (\upsilon >>> i) r^T,$$

де $\upsilon = (01, 01, 05, 01, 08, 06, 07, 04)$, $>>> i$ – операція циклічного зсуву на i розрядів праворуч, $i = 0, 1, \ldots, 7$,

$$r^T = \left(r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7\right)^T.$$

4. Виводиться вихідне значення q, яке інтерпретується як 64-бітовий рядок.

Швидке обчислення вектору $\left(q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\right) = Q$ реалізується за правилом

$$Q^T = T_0[w_0] \oplus T_1[w_1] \oplus T_2[w_2] \oplus T_3[w_3] \oplus T_4[w_4] \oplus T_5[w_5] \oplus T_6[w_6] \oplus T_7[w_7],$$

де

$$T_0[a] = \begin{pmatrix} 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \end{pmatrix} \cdot \pi_0\left[a\right],\ T_1[a] = \begin{pmatrix} 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \end{pmatrix} \cdot \pi_1\left[a\right],\ T_2[a] = \begin{pmatrix} 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \end{pmatrix} \cdot \pi_2\left[a\right],\ T_3[a] = \begin{pmatrix} 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \end{pmatrix} \cdot \pi_3\left[a\right],$$

$$T_4[a] = \begin{pmatrix} 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \end{pmatrix} \cdot \pi_0\left[a\right],\ T_5[a] = \begin{pmatrix} 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \end{pmatrix} \cdot \pi_1\left[a\right],\ T_6[a] = \begin{pmatrix} 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \end{pmatrix} \cdot \pi_2\left[a\right],\ T_7[a] = \begin{pmatrix} 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \end{pmatrix} \cdot \pi_3\left[a\right].$$

Застосування таблиць-констант $T_i[a]$, $i = 0,1,...,7$ дозволяє значно зменшити кількість операцій, зокрема функція нелінійної підстановки обчислюється за сім операцій XOR над 64-бітовими рядками.

Множення на α в арифметиці поля $GF(2^{64})$

Множення на α в арифметиці поля $GF(2^{64})$ реалізується за допомогою таблиці передобчислень Mul_α з 256 рядків по 64 бітів в кожному.

Вхід: 64-бітовий рядок w, що представляє елемент поля $GF(2^{64})$.

Вихід: 64-бітовий рядок $w' = w \otimes \alpha$, що представляє елемент поля $GF(2^{64})$.

1. Обчислюється значення

$$w' = (w << 8) \oplus Mul_\alpha[w >> 56], \qquad (1)$$

де $w << 8$ є результатом зсуву ліворуч (в бік старших розрядів) 64-бітового рядка w на вісім розрядів із заповненням молодших розрядів нульовими значеннями;

$w >> 56$ є результатом зсуву праворуч (в бік молодших розрядів) 64-бітового рядка w на 56 розрядів із заповненням старших розрядів нульовими значеннями. Вісім молодших розрядів вектору $w >> 56$ інтерпретуються як елемент поля $GF(2^8)$ для індексації таблиці передобчислень Mul_α;

Mul_α – таблиця-константа з 256 рядків по 64 біта в кожному (таблиця передобчислень);

$Mul_\alpha[c]$ – 64-бітне значення таблиці передобчислень у рядку з індексом c, де c представляє елемент поля $GF(2^8)$, $Mul_\alpha[c]$ представляє елемент поля $GF(2^{64})$.

2. Виводиться вихідне значення w'.

Множення на α^{-1} в арифметиці поля $GF(2^{64})$

Множення на α^{-1} в арифметиці поля $GF(2^{64})$ реалізується за допомогою таблиці передобчислень $Mul_{\alpha^{-1}}$ з 256 рядків по 64 бітів в кожному.

Вхід: 64-бітовий рядок w, що представляє елемент поля $GF(2^{64})$.

Вихід: 64-бітовий рядок $w' = w \otimes \alpha^{-1}$, що представляє елемент поля $GF(2^{64})$.

1. Обчислюється значення

$$w' = (w >> 8) \oplus Mul_{\alpha^{-1}}[w \,\&\, \gamma], \tag{2}$$

де $w >> 8$ є результатом зсуву праворуч (в бік молодших розрядів) 64-бітового рядка w на 8 розрядів із заповненням старших розрядів нульовими значеннями;

$w \,\&\, \gamma$ є результатом побітової кон'юнкції 64-бітового рядка w та 64-бітового рядка γ, який у шістнадцятковому поданні має вигляд $\gamma = 00000000000000FF$. Вісім молодших розрядів вектору $w \,\&\, \gamma$ інтерпретуються як елемент поля $GF(2^8)$ для індексації таблиці передобчислень $Mul_{\alpha^{-1}}$;

2. Виводиться вихідне значення w'.

Значення таблиць-констант Mul_α, $Mul_{\alpha^{-1}}$

Для швидкого шифрування застосовуються таблиці передобчислень Mul_α, $Mul_{\alpha^{-1}}$. Це дозволяє значно зменшити кількість операцій для обробки блоку вхідних даних.

Поліном, що задає зворотний зв'язок LFSR має вигляд $f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha$, де α і α^{-1} належать полю $GF(2^{64})$, причому $\alpha = z$ є коренем примітивного над полем $GF(2^8)$ поліному $g(z)$. Таким чином, у кожній комірці LFSR зберігається 64-бітна послідовність w, яку представимо у вигляді восьми підблоків w_j по вісім біт у кожному:

$$w = (w_7, w_6, w_5, w_4, w_3, w_2, w_1, w_0),$$

які інтерпретуються як коефіцієнти поліному

$$w(z) \in GF(2^8)[z] / (g(z)).$$

Якщо $\alpha = z$ є коренем примітивного над $GF(2^8)$ поліному $g(z) = z^8 + g_7 z^7 + ... + g_1 z + g_0$, тоді маємо:

$$\begin{aligned} w(z) \cdot \alpha &= (w_7 z^8 + w_6 z^7 + ... + w_1 z^2 + w_0 z) \bmod g(z) \equiv \\ &\equiv (w_6 + w_7 g_7) z^7 + (w_5 + w_7 g_6) z^6 + ... + (w_0 + w_7 g_1) z + (w_7 g_0) z^0 = \\ &= w_{<<8}(z) + w_{>>56}(z) \cdot g'(z), \end{aligned}$$

де поліноми $w_{<<8}(z)$, $w_{>>56}(z)$ та $g'(z)$ мають вигляд:

$$w_{<<8}(z) = w_6 z^7 + w_5 z^6 + ... + w_0 z,$$

$$w_{>>56}(z) = w_7,$$

$$g'(z) = g_7 z^7 + g_6 z^6 + ... + g_1 z + g_0.$$

Двійкове подання коефіцієнтів поліномів $w_{<<8}(z)$ і $w_{>>56}(z)$ утворює розглянуті у (1) двійкові послідовності $w << 8$ і $w >> 56$. Отже, обчислення $w(z) \cdot \alpha$ в арифметиці поля

$GF(2^{64})$ відповідає формулі (1), де 256 значень таблиці $Mul_{\alpha}[w_7]$ розраховуються як 64-бітні послідовності при двійковому поданні коефіцієнтів $(w_7g_7, w_7g_6, ..., w_7g_1, w_7g_0)$ поліному

$$w_{>>56}(z)\cdot g'(z) = w_7(g_7z^7 + g_6z^6 + ... + g_1z + g_0)$$

для кожного з 256 можливих значень $w_7 \in GF(2^8)$.

Якщо $\alpha = z$ є коренем примітивного над $GF(2^8)$ поліному $g(z) = z^8 + g_7z^7 + ... + g_1z + g_0$, тоді маємо:

$$\alpha^8 = g_7\alpha^7 + ... + g_1\alpha + g_0\alpha^0,$$

або

$$\alpha^7 = g_7\alpha^6 + ... + g_1\alpha^0 + g_0\alpha^{-1},$$

отже

$$g_0^{-1}\alpha^7 + g_0^{-1}g_7\alpha^6 + ... + g_0^{-1}g_1\alpha^0 = \alpha^{-1} = z^{-1}.$$

Тоді

$$w(z)\alpha^{-1} = w_7z^6 + w_6z^5 + ... + w_1z^0 + w_0z^{-1} =$$
$$w_7z^6 + w_6z^5 + ... + w_1z^0 + w_0(g_0^{-1}z^7 + g_0^{-1}g_7z^6 + ... + g_0^{-1}g_1z^0) =$$
$$= (w_0g_0^{-1})z^7 + (w_7 + w_0g_0^{-1}g_7)z^6 + ... + (w_1 + w_0g_0^{-1}g_1)z^0 =$$
$$= w_{>>8}(z) + w_0(z)\cdot g''(z),$$

де поліноми $w_{>>8}(z)$, $w_0(z)$ та $g''(z)$ мають вигляд:

$$w_{>>8}(z) = w_7z^6 + w_6z^5 + ... + w_2z + w_1,$$
$$w_0(z) = w_0,$$
$$g''(z) = g_0^{-1}z^7 + g_0^{-1}g_7z^6 + ... + g_0^{-1}g_2z + g_0^{-1}g_1.$$

Двійкове подання коефіцієнтів поліномів $w_{>>8}(z)$ і $w_0(z)$ утворює розглянуті у (2) двійкові послідовності $w>>8$ і γ. Таким чином, обчислення $w(z)\cdot\alpha^{-1}$ в арифметиці поля $GF(2^{64})$ відповідає формулі (2), де 256 значень таблиці $Mul_{\alpha^{-1}}[w_0]$ розраховуються як 64-бітні послідовності при двійковому поданні коефіцієнтів $(w_0g_0^{-1}, w_0g_0^{-1}g_7, ..., w_0g_0^{-1}g_2, w_0g_0^{-1}g_1)$ поліному

$$w_0(z)\cdot g''(z) = w_0(g_0^{-1}z^7 + g_0^{-1}g_7z^6 + ... + g_0^{-1}g_2z + g_0^{-1}g_1)$$

для кожного з 256 можливих значень $w_0 \in GF(2^8)$.

Дослідження швидкості програмної реалізації

Для дослідження швидкості формування ключового потоку ми реалізували в рівних умовах найбільш відомі симетричні криптоперетворення. Список алгоритмів, джерело специфікації та короткі відомості наведено у табл. 1.

Таблиця 1

Криптоалгоритми, обрані для порівняння

Шифр	Джерело специфікації	Розмір стану, біт	Розмір ключа, біт	Розмір IV, біт
AES	FIPS-197, CRYPTREC, ISO/IEC 18033-4	128	128, 256	256
Калина	ДСТУ 7624:2014	128, 256, 512	128, 256, 512	128, 256, 512
HC	eSTREAM	128, 256	128, 256	128, 256

MICKEY	eSTREAM	160	128	128
RABBIT	ISO/IEC 18033-4, eSTREAM	513	128	64
SALSA-20	eSTREAM	512	128	64
SNOW2.0	ISO/IEC 18033-4	512	128, 256	128, 56
SOSEMANUK	eSTREAM	512	128	128
TRIVIUM	eSTREAM, ISO/IEC 29192-3	288	80	80
Enocoro	ISO/IEC 29192-3	272	80, 128	64
CRYPTMT3	eSTREAM	128	128	64
DECIMv2	ISO/IEC 18033-4, eSTREAM	288	128	128
RC4	Список розсилки Cypherpunks	256	256	–
KCIPHER-2	ISO/IEC 18033-4, CRYPTREC	640	128	128
GRAIN	eSTREAM	128	128	96
MUGI	ISO/IEC 18033-4	128	128	128
Струмок-256	Цей документ	1024	256	256
Струмок-512		1024	512	512

Результати тестування за критерієм шифрування довгих потоків [20] наведено в табл. 2.

Таблиця 2

Результати оцінки швидкодії шифрів

Шифр	Intel Core i7-6820HQ 2.7Gh	Intel Core i7-5500u 2.4Gh	Intel Pentium P6200 2.13Gh
AES-128	2,48	1,75	1,12
AES-256	1,66	1,18	0,80
Калина-128	2,56	1,79	0,83
Калина-256	1,71	1,21	0,57
Калина-512	1,42	0,99	0,46
HC-128	11,46	7,69	4,25
HC-256	5,13	3,88	2,03
MICKEY-128	0,07	0,05	0,03
RABBIT	3,65	2,77	1,64
SALSA-20	3,02	2,06	1,41
SNOW2.0-128	8,76	5,43	3,67
SNOW2.0-256	8,72	5,54	3,59
SOSEMANUK	4,07	2,56	1,82
TRIVIUM	3,89	2,78	1,89
CryptMT3	5,92	4,63	4,04
DECIM-128	0,01	0,01	0,01
RC4	3,58	3,21	1,67
KCIPHER-2	0,40	0,40	0,31
GRAIN	0,01	0,01	0,00
MUGI	3,62	2,98	2,58
Струмок-256	13,31	10,04	5,10
Струмок-512	13,70	9,74	5,08

Як видно із даних таблиці, генератор ключових потоків *Струмок* дозволяє формувати псевдовипадкові послідовності із швидкістю понад 10 Гбіт/с. За цим показником він випереджає майже всі найбільш поширені шифри, зокрема і алгоритм *SNOW2.0*.

Висновки

Генератор ключового потоку *Струмок* у своїй концептуальній схемі подібний до SNOW2.0. Але розробники SNOW2.0 зосереджувалися на використанні 32-розрядних обчислювальних систем, тоді як *Струмок* призначений для використання в більш потужних 64-розрядних обчислювальних системах. У зв'язку з цим в алгоритмі *Струмок* підвищується швидкість формування псевдовипадкової послідовності, що використовується 64-розрядними словами, для зберігання потоку ключів шифрування. Проведені порівняльні тести показали, що алгоритм *Струмок* на 32-розрядних обчислювальних системах також демонструє хороші результати роботи. Використання попереднього обчислення збільшує швидкість алгоритму, оскільки в процесі генерації ключові потоку немає необхідності в складних калькуляціях.

В алгоритмі *Струмок* збільшені, в порівнянні з *SNOW2.0*, довжини секретного ключа та вектору ініціалізації. Це дозволяє надійно застосовувати потоковий шифр навіть з із врахуванням квантових методів криптографічного аналізу. Отже за сукупністю властивостей *Струмок* може розглядатися у якості кандидата на національний стандарт симетричного шифрування в Україні.

Список литературы:

1. Ferguson N. and Schneier B. Practical Cryptography. John Wiley & Sons. 2003. 432 p.
2. Menezes A.J., P.C. van Oorschot, Vanstone S.A. Handbook of Applied Cryptography. CRC Press, 1997, 794 p.
3. Koblitz N. and Menezes A.J. A Riddle Wrapped in an Enigma. Internet: https://eprint.iacr.org/2015/1018.pdf, Oct. 20, 2015 [Aug. 21, 2016].
4. Bernstein D., Buchmann J. and Dahmen E.. Post-Quantum Cryptography. Springer-Verlag, Berlin-Heidleberg, 2009, 245 p.
5. Moody D. Post-Quntum Cryptography: NIST's Plan for the Future // The Seventh International Conference on Post-Quntum Cryptography, Japan, 2016. [On-line]. Internet: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf [March 8, 2016].
6. ISO/IEC 18033-4:2011. Information technology – Security techniques – Encryption algorithms – Part 4: Stream ciphers. On-line]. Internet: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54532 [Dec., 2012].
7. Kuznetsov O., M. Lutsenko and D. Ivanenko, "Strumok stream cipher: Specification and basic properties," 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 59-62.
8. Kuznetsov O., Gorbenko Y. and Kolovanova I. Combinatorial properties of block symmetric ciphers key schedule, 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 55-58.
9. Gorbenko I., Kuznetsov A., Lutsenko M. and Ivanenko D. The research of modern stream ciphers // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkov, 2017, pp. 207-210.
10. Kuznetsov A., Svatovskij I., Kiyan N. and Pushkar'ov A. Code-based public-key cryptosystems for the post-quantum period // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkov, 2017, pp. 125-130.
11. A. Kuznetsov, I. Kolovanova and T. Kuznetsova. Periodic characteristics of output feedback encryption mode // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkov, 2017. pp. 193-198.
12. Kuznetsov O., Gorbenko Y., Andrushkevych A. and Belozersev I. Analysis of block symmetric algorithms from international standard of lightweight cryptography ISO/IEC 29192-2. 2017 // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). Kharkov, 2017. pp. 203-206.
13. Izbenko Y., Kovtun V. and Kuznetsov A. The Design of Boolean Functions by Modified Hill Climbing Method // Sixth International Conference on Information Technology: New Generations, Las Vegas, NV, 2009, pp. 356-361.
14. Kuznetsov A., Serhiienko R. and Prokopovych-Tkachenko D. Construction of cascade codes in the frequency domain // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). Kharkov, 2017. pp. 131-136.
15. Andrushkevych A., Kuznetsova T., Bilozertsev I. and Bohucharskyi S. The block symmetric ciphers in the post-quantum period // Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), Kharkiv, 2016, pp. 43-46.

16. Gorbenko I.D., Dolgov V.I., Rublinetskii V.I., Korovkin K.V. Methods of Information Protection in Communications Systems and Methods of Their Cryptoanalysis // Telecommunications and Radio Engineering. 1998. Volume 52, Issue 4, pp. 89-96.

17. Gorbenko I., Ponomar V. Examining a possibility to use and the benefits of post-quantum algorithms dependent on the conditions of their application // EasternEuropean Journal of Enterprise Technologies. Vol 2, No 9 (86) (2017), pp. 21-32.

18. Stasev Yu.V., Kuznetsov A.A. Asymmetric code-theoretical schemes constructed with the use of algebraic geometric codes // Kibernetika i Sistemnyi Analiz, No. 3, pp. 47-57, May-June 2005.

19. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. [On-line]. Internet: http://shop.uas.org.ua/ua/informacijni-tehnologii-kriptografichnij-zahist-informacii-algoritm-simetrichnogo-blokovogo-peretvorennja.html

20. eSTREAM Optimized Code HOWTO. [On-line]. Internet: http://www.ecrypt.eu.org/stream/perf/ [Nov. 1, 2005].

Харківський національний
університет імені В.Н. Каразіна

Надійшла до редколегії 12.02.2018