УДК 004.056.55

М.В. Лемешко[1]
В.С. Мусій[2]

Покращення алгоритму шифрування HC-128

Вінницький національний технічний університет

Анотація

Проведено огляд алгоритму шифрування HC-128, виконані аналітичні, та практичні дії для підвищення крипостійкості алгоритму від зовнішнього втручання.

Ключові слова: *HC-128, шифрування, функції, алгоритми, робота з масивами*

Abstract

An overview of the encryption algorithm HC-128 was carried out, analytical and practical actions were performed to increase the cryptographic stability of the algorithm from external interference.

Keywords: *HC-128, encryption, functions, algorithms, work with arrays*

У всьому різноманітті проблем забезпечення інформаційної безпеки, розв'язуваних за допомогою криптографічних методів і засобів, задача захисту мікропроцесорів є на сьогоднішній день однією з найгостріших. З урахуванням сучасних вимог до інформаційно-комунікаційних систем ця задача все частіше і частіше перетворюється в серйозну проблему.

HC-128 ─ простий, безпечний, програмно-ефективний шифр з відкритим вихідним кодом. Потоковий шифр HC-128 є спрощеною версією HC-256 [1] для 128-бітної безпеки. Зручність використання цього алгоритму в організації захисту мікропроцесорів забезпечує велику кількість споживачів, наражених на небезпеку.

Алгоритм не перевантажений довгими масивами констант і використовує лише 6 основних функцій:

1. f1(x) = (x >>> 7) (x >>> 18) (x >> 3)
2. f2(x) = (x >>> 17) (x >>> 19) (x >> 10)
3. q1(x,y,z) = ((x >>> 10) (z >>> 23)) + (y >>> 8)
4. q1(x,y,z) = ((x <<<10) (z <<< 23)) + (y <<< 8)
5. h1(x) = Q[x0] + Q[256 + x2]
6. h2(x) = P[x0] + P[256 + x2], де

⊕: бітове виключення
>>: оператор зсуву вправо на вказану кількість біт
<<: оператор зсуву вліво на вказану кількість біт
>>>: оператор правого повороту. x >>> n означає ((x >> n) ⊕ (x << (-n))
<<<: оператор лівого повороту. x <<< n означає ((x << n) ⊕ (x >> (-n))
P,Q: таблиця з 1024-х 32-бітних елементів.

Функції алгоритму HC-128

Функції f1, f2 (зображено на рис.1) являють собою однозначне перетворення одного 32-ух бітного числа в інше. Тобто, на кожному раунді черговий шматок рядка вбудовується тільки до частини стану, тоді як псевдо-випадкова перестановка f обробляє весь стан цілком, розчиняючи таким чином рядок стана і роблячи його залежним від всього рядка, що безпосередньо впливає на загальний стан захищенності алгоритму. [2]

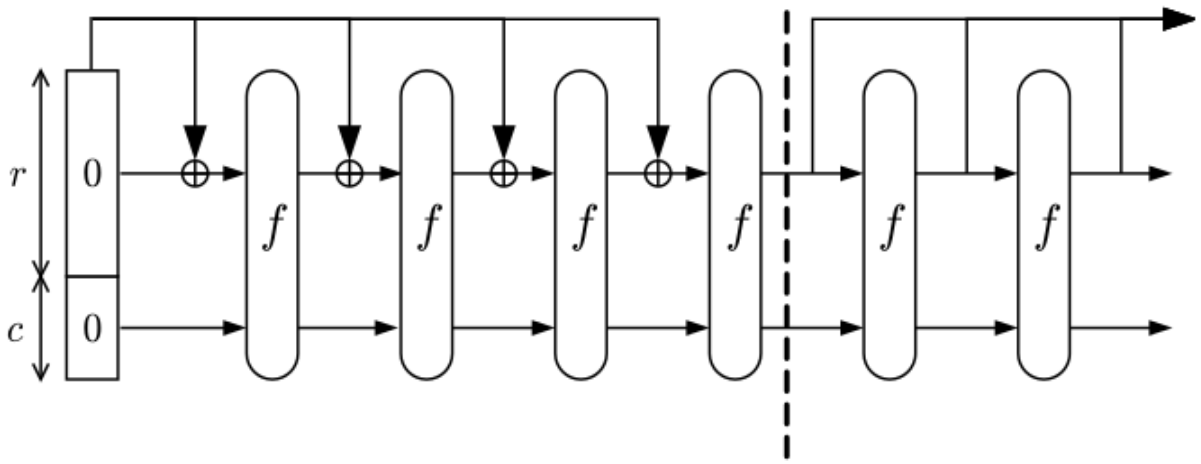

Рисунок 1 – візуальне відображення роботи алгоритму функцій f1,f2

Форми h1, h2 необхіжно модифікувати таким чином, щоб включити усі 32-а біти власних входів. У запропонованій версії цих функцій використовується XOR-введення з існуючим виходом (сума двох записів масиву)[3]. Використовуючи константи які мають однозначну відповідність (0 - (2^{32}-1)) і більший цикл отримаємо напорядок кращі результати:

```
private static int f1(int x)
  {
    return rotateRight(x, 22) ^ rotateRight(x, 13) ^ (x >>> 3);
  }
  private static int f2(int x)
  {
    return rotateRight(x, 18) ^ rotateRight(x, 4) ^ (x >>> 9);
  }
```

Функції q1 (x) і q2 (x) являють собою масиви стану P і Q, та виконуються незалежно один від одного. Тому при несприятливих умовах (перетинання з одним із цих масивів і частин потоку згенерованих байтів) це може привести до несприятливих наслідків. [4]
Необхідно модифікувати функції q1 і q2 таким чином, щоб кожен елемент P залежав від випадкового елемента Q і навпаки. Використаємо замість циклічного переміщення елементу кілька біт у якості індексу для елемента з масивів Q і P. Необхідно змінити функції оновлення q1 і q2 для збереження внутрішнього стану, а також для того, щоб забезпечити випадковість ключа:

```
private int g1(int x, int y, int z)
  {
    return (rotateRight(x, 10) ^ rotateRight(z, 23)) + Q[(y >> 7) & 0x1FF];
  }
  private int g2(int x, int y, int z)
  {
    return (rotateLeft(x, 10) ^ rotateLeft(z, 23)) + P[(y >> 7) & 0x1FF];
  }
```

Висновки: Аналіз алгоритму показав, що існуючий метод захисту не здатен виконувати свої функції через помилки проектування. Алгоритм виявився відкритим до зовнішнього втручання шляхом вбудовування інформації між даними масивів.

Проведені роботи з покращення алгоритму гарантують, що кожен новий блок масиву P (або Q) залежить від попереднього блоку Q (або P) масиву, який не буде застосовано єдиним масивом і внутрішній стан буде збережений, навіть якщо половина внутрішніх елементів буде відома.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ложников П.С. Биометрическая система аутентификации пользователя по динамике подписи // Материалы V Всерос. науч.-практ. конф. «Проблемы информационной безопасности государства, общества и личности». Томск: ТУСУР, 2013. – С. 134-135.
2. Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М.: ДМК Пресс, 2016. 592 с.
3. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. Новосибирск: Изд-во Ин-та математики, 2017. 270 с.
4. Shivani Hashiaa, Chris Pollettb, Mark Stampc, ON USING MOUSE MOVEMENTS AS A BIOMETRIC. Dept. of Computer of Science, MacQuarrie Hall, San Jose State University, One Washington Square, San Jose, CA 95195, USA 2014.

Лемешко Максим Володимирович – студент групи УБ-17м, Кіберзбезпека, Вінницький національний технічний університет, Вінниця, e-mail: MaxKondor227@gmail.com.

Мусій Владислав Сергійович – студент групи УБз-17м, Кіберзбезпека, Вінницький національний технічний університет, Вінниця, e-mail: vladmusii24@gmail.com.

Науковий керівник: ***Ткачук Людмила Миколаївна*** – к.е.н, доцент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет, Вінниця.

Lemeshko Maxim - student group UB17m, Cyber Security, Vinnytsia National Technical University, Vinnytsia, e-mail: max.lemeshko@gmail.com.

Musii Vladislav - student group UBr17m, Cyber Security, Vinnytsia National Technical University, Vinnytsia, e-mail: vladmusii24@gmail.com.

Scientific supervisor: ***Lyudmila Mykolayivna Tkachuk*** – Candidate of Economic Sciences, Associate Professor of the Department of Management and Security of Information Systems, Vinnytsia National Technical University, Vinnytsia.