Література

1. Устенко С. В. Моделювання процесів функціонування та розвитку наукомістких виробничих систем : дис. докт. техн. наук : 08.00.11 / Устенко Станіслав Веніамінович — ДВНЗ «Київський національний економічний університ імені Вадима Гетьмана», 2008.
2. Каргін Б. Б. Впровадження інноваційних інформаційних технологій у діяльність промислових підприємств : 08.00.04 / Каргін Борис Борисович — Приазовський державний технічний університет ДВНЗ, 2019.

References

1. Ustenko S.V. Modeling of processes of functioning and development of high-tech production systems: diss. doc. tech. Sciences: 08.00.11 / Ustenko Stanislav Veniaminovich — DVNZ «Kyiv National Economic University, 2008.
2. Kargin B.B. Implementation of innovative information technologies in the activities of industrial enterprises: 08.00.04 / Kargin Boris Borisovich — Azov State Technical University of DVNZ, 2019.

Статтю подано до редакції 07.02.2019 р.

УДК 004.4 DOI: 10.33111/mise.97.9

Гладка Ю. А., к.ф.-м.н.,
доцент кафедри комп'ютерної математики та інформаційної безпеки
Щедріна О. І., к.е.н.,
доцент кафедри інформаційного менеджменту
Загорний І. Р.,
студент 3-го курсу спеціальності «Кібербезпека»,
Київський Національний економічний університет
імені Вадима Гетьмана

Gladka Y. A., PhD in Physics and Mathematics,
Associate Professor of the
Computer Mathematics and Information Security Department,
Shchedrina O. I., PhD in Economics,
Associate Professor of the Information Management Department,
Zagorniy I. R.,
3rd year Student at the «Cybersecurity» speciality,
Kyiv National Economic University named after Vadym Hetman

НОВИЙ МЕТОД ШИФРУВАННЯ ADIANTIUM

ADIANTIUM AS A NEW ENCRYPTION METHOD

Анотація. *Для багатьох застосувань для шифрування зберігання шифротекст повинен бути такого ж розміру, як і простий текст;*

загалом це відповідає розміру дискового сектору або 512, або 4096 байт. Це означає, що не можна застосовувати стандартні підходи, такі як AES-GCM або RFC7539. Стандартне рішення — AES-XTS, але це має два недоліки: якщо апаратне забезпечення AES відсутнє, AES є відносно повільним, особливо впровадження в постійний час. Використовуючи XTS, однобітна зміна в простому тексті означає лише 16-байтову зміну на шифротекст, що виявляє більше для нападника, ніж потрібно.

Розробники Google представили новий метод шифрування Adiantium, який орієнтований на бюджетні пристрої, де використання AES неможливо.

Справа в тому, що користувачам Android доступна підтримка алгоритму шифрування AES (Advanced Encryption Standard), який відмінно працює з новітніми процесорами за рахунок ARMv8 Cryptography Extensions. Однак на менш потужних пристроях, починаючи від бюджетних смартфонів і закінчуючи «розумними» годинами і телевізорами, починаються проблеми. Такі гаджети оснащені менш потужними процесорами, де апаратної підтримки AES «з коробки» просто немає (наприклад, ARM Cortex-A7).

Інженери Google пояснюють, що на таких пристроях AES працює настільки повільно, що це псує користувачеві весь досвід взаємодії з пристроєм. І хоча шифрування сховища стало обов'язковою умовою ще в 2015 році, з релізом Android 6.0, малопотужні пристрої були «звільнені» від цього, так як при включенні AES вони більше гальмують, ніж працюють. У таких випадках шифрування або відключено за замовчуванням, щоб уникнути проблем, або взагалі видалено з Android.

Саме для таких пристроїв з малопотужними процесорами розробники Google і створили Adiantum, що працює з потоковим шифром ChaCha20.
Ключові слова: *криптографія, ключ, AES, апаратне забезпечення, шифрування.*

Abstract. *For many storage encryption applications, the ciphertext must be the same size as the plaintext; generally this matches the disk sector size of either 512 or 4096 bytes. This means that standard approaches like AES-GCM or RFC7539 cannot be applied. The standard solution is AES-XTS, but this has two disadvantages: if AES hardware is absent, AES is relatively slow, especially constant-time implementations. Using XTS, a one-bit change to the plaintext means only a 16-byte change to the ciphertext, revealing more to the attacker than necessary.*

Google's developers have introduced a new Adiantium encryption method that targets low-cost devices where AES cannot be used.

The fact is that Android users have the support of AES (Advanced Encryption Standard) encryption algorithm, which works well with the latest processors through ARMv8 Cryptography Extensions. However, on less powerful devices, from budget smartphones to smart watches and TVs, problems begin. Such gadgets are equipped with less powerful processors, where AES hardware out of the box simply does not exist (for example, ARM Cortex-A7).

Google engineers explain that such AES devices run so slowly that it robs the user of the whole experience of interacting with the device. And although storage encryption became a prerequisite in 2015, with the release of Android 6.0, low-power devices were «released» from this, since when AES was turned on, they slowed down more than they did. In such cases, encryption is either disabled by default to avoid problems or removed from Android altogether.

It is for such devices with low-power processors that Google developers have created Adiantum that works with the ChaCha20 streaming encryption.
Keywords: *cryptography, key, AES, hardware, encryption.*

Вступ: Комп'ютерна криптографія (з 70-х років XX століття) зобов'язана своєю появою обчислювальним засобам з продуктивністю, достатньою для реалізації кріптосистем, що забезпечують при великій швидкості шифрування на кілька порядків вищу криптостойкость, ніж «ручні» і «механічні» шифри.

Першим класом криптосистем, практичне застосування яких стало можливо з появою потужних і компактних обчислювальних засобів, стали блокові шифри. У 70-і роки був розроблений американський стандарт шифрування DES (прийнятий в 1978 році). Один з його авторів, Хорст Фейстел (співробітник IBM), описав модель блокових шифрів, на основі якої були побудовані інші, більш стійкі симетричні криптосистеми, в тому числі вітчизняний стандарт шифрування ГОСТ 28147-89.

З появою DES збагатився і криптоаналіз, для атак на американський алгоритм був створено кілька нових видів криптоаналізу (лінійний, диференціальний і т.д.), практична реалізація яких знову ж була можлива тільки з появою потужних обчислювальних систем.

У 80–90-ті роки з'явилися зовсім нові напрямки криптографії: розподіл усіх шифрування, квантова криптографія та інші. Усвідомлення їх практичної цінності ще попереду. Актуальною залишається і завдання вдосконалення симетричних криптосистем. У 80–90-х роках були розроблені нефейстеловскіе шифри (SAFER, RC6 і ін.), а в 2000 році після відкритого міжнародного конкурсу був прийнятий новий національний стандарт шифрування США — AES.

Постановка проблеми: В сучасні часи сформувалась потреба шифрування файлів на системах Android, для цього винайшли та використовують метод шифрування Adiantium, оскільки воно захищає данні, при втраті мобільного пристрою.

Шифрування Andriod використовує покращений стандарт шифрування (AES), та більшість сучасних процесорів, на базі Armv8, постачаються с криптоприскорювачем, який збільшує виробничість у кілька разів на відміну від програмного рішення. Але також є багато бюджетних пристроїв, в яких відсутні криптографічні рішення та включення шифрування AES зробить пристрій ще повільніше, ніж такими як вони є.

Виклад основного матеріалу: Adiantum — це шифрова конструкція для шифрування диска, яка використовує шифри ChaCha та Advanced Encryption Standard (AES) і код аутентифікації криптографічних повідомлень Poly1305 (MAC).

Він був розроблений у 2018 році Полом Кроулі та Еріком Біггерсом у Google спеціально для мобільних пристроїв з низьким рівнем живлення, на яких працює Android Go. Він включений в ядро Linux з версії 5.0. HPolyC — це більш ранній варіант Adiantum, який використовує іншу конструкцію для хеш-функції Poly1305.

Adiantum реалізований в Android 10 як альтернативний шифр для шифрування пристроїв, особливо на пристроях низького класу, у яких відсутні апаратно-прискорена підтримка AES. Компанія заявила, що Adiantum працює в п'ять разів швидше, ніж AES на процесорах ARM Cortex-A7. Раніше компанія Google звільняла пристрої від обов'язкового шифрування, якщо їх технічні характеристики впливали на продуктивність системи, якщо вони включені. Завдяки впровадженню Adiantum шифрування пристрою стає обов'язковим для всіх пристроїв Android, починаючи з Android 10.

За аналогією з AES-128-CBC-ESSIV і AES-XTS метод Adiantum не змінює результуючий розмір даних, що дозволяє використовувати його для шифрування секторів на накопичувачах. Adiantum також забезпечує генерацію блоків з різним шіфротекста для повторюваних вихідних даних. Реалізація Adiantum базується на застосуванні швидкої хеш-функції NH, алгоритмі аутентифікації повідомлень (MAC) Poly1305 і потоковому шифрі XChaCha12, а також одноразової операції на базі блокового шифру AES-256 для 16 байт у кожному блоці (з урахуванням розміру блоку в 4096 байт така операція не критична з точки зору продуктивності).

Poly1305 і XChaCha12 позиціонуються як швидші і безпечніші аналоги HMAC і AES, програмна реалізація яких дозволяє домогтися фіксованого часу виконання без задіяння спеціального апаратної підтримки. Для підвищення продуктивності алгоритм ChaCha застосовується у варіанті з 12 раундами замість зазвичай використовуваних 20, але цього цілком достатньо, так як ChaCha навіть з 12 раундами забезпечує вищий рівень стійкості до атак, ніж AES-256. На процесорі ARM Cortex-A7 реалізація Adiantum витрачає на операцію розшифровки 10.6 циклів процесора на кожен байт (при розмірі блоку 4096 байт, рис. 1), що в п'ять разів швидше AES-256-XTS.

На процесорах з апаратною підтримкою прискорення AES, таких як ARMv8 з інструкціями A64, A32 і T32 (Cryptography Extensions) і x86 з інструкціями AES-NI, рекомендується застосовувати систему шифрування дисків на базі AES, так як у цьо-

му випадку апаратно прискорений AES буде швидше програмної реалізації Adiantum. При цьому Adiantum забезпечує вищу стійкість до атак, так як в AES-XTS зміна одного байта вихідних даних призводить до зміни всього 16 байт шифротекста, в той час як в Adiantum змінюється цілком весь блок, рівний розміру сектора (512 або 4096 байт).

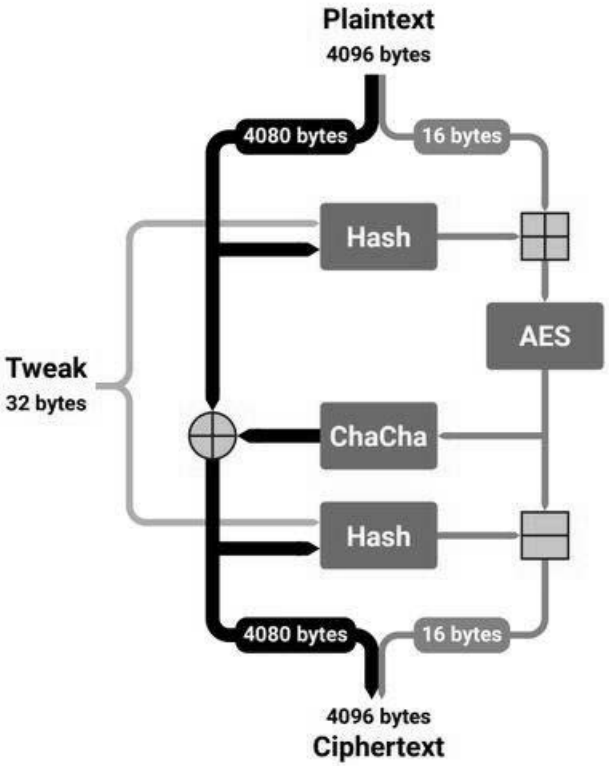

Рис. 1. Принцип шифрування Adiantium

Salsa20 — система поточного шифрування, розроблена Даніелем Бернштейном. Алгоритм був представлений на конкурсі «eSTREAM», метою якого було створення європейських стандартів для шифрування даних, що передаються поштовими системами. Алгоритм став переможцем конкурсу в першому профілі (потокові шифри для програмного застосування з великою пропускною здатністю).

Шифр Salsa20 використовує такі операції:

- складання 32-бітних чисел;
- побітове додавання по модулю 2 (xor);
- зрушення бітів.

Алгоритм використовує хеш-функцію з 20 циклами. Основні її перетворення нагадують алгоритм AES.

Далі будемо називати кожен елемент множини $\{0,1, ..., 2^{32} - 1\}$ і записувати в шістнадцятковому вигляді з префіксом 0_x.

Операцію складання двох слів по модулю 2^{32} будемо позначати знаком «+». Що виключає або (побітовое підсумовування) будемо позначати символом «$\oplus$», с — бітовий циклічний лівий зсув слова

u будемо позначати u $\lll$ c. Якщо u уявити як $u = \sum_{i=0} 2^i u_i$, тоді

$$u \lll c = \sum_{i=0} 2^{i+c \bmod 32} u_i. \quad (1)$$

Основним блоком системи є перетворення quarterround(y) над чотирма словами. З нього будуються далі описані загальніші перетворення.

Його суть полягає в тому, що для кожного слова ми складаємо два попередніх, зрушуємо (циклічно) суму на певну кількість біт і побитово підсумовуємо результат з обраним словом. Наступні операції проводяться з новими значеннями слів.

Припустимо, що y — послідовність 4 слів $y = (y_0, y_1, y_2, y_3)$ тоді функція quarterround(y) $= (z_0, z_1, z_2, z_3)$ де

$$z_1 = y_1 \oplus ((y_0 + y_3) \lll 7),$$
$$z_2 = y_2 \oplus ((z_1 + y_0) \lll 9),$$
$$z_3 = y_3 \oplus ((z_2 + z_1) \lll 13),$$
$$z_0 = y_0 \oplus ((z_3 + z_2) \lll 18).$$

Наприклад:

Quarterround (0x00000001; 0x00000000; 0x00000000; 0x00000000) = (0x08008145; 0x00000080; 0x00010200; 0x20500000)

Можна розглядати цю функцію як перетворення слів y_0, y_1, y_2, y_3 (рис. 2). Кожне з таких перетворень можна зупинити, як і функція в цілому.

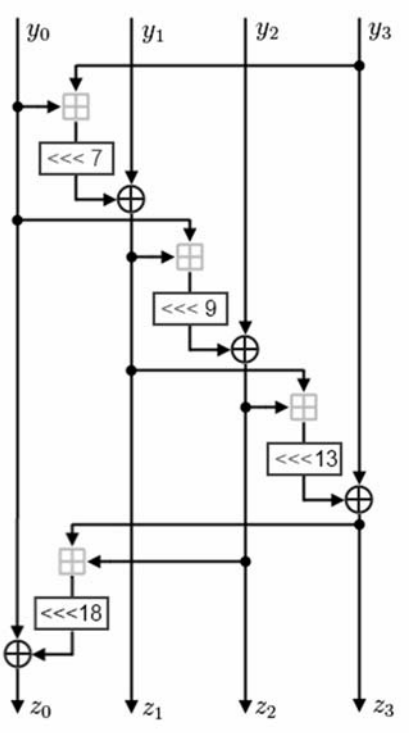

Рис. 2. Перетворення слів

rowround (y) $$y = \begin{pmatrix} y_0 & y_1 & y_2 & y_3 \\ y_4 & y_5 & y_6 & y_7 \\ y_8 & y_9 & y_{10} & y_{11} \\ y_{12} & y_{13} & y_{14} & y_{15} \end{pmatrix} \quad (1)$$

У цьому перетворенні ми беремо 16 слів. Уявімо їх у вигляді матриці 4x4. Беремо кожен ряд цієї матриці і перетворимо слова цієї матриці функцією quarterround(y). Слова з рядка беруться по порядку, починаючи з і-го для *i*-го рядка, де *i* = {0,1,2,3}.

Нехай y = (y_0, y_1, y_2,… y_{15}) — послідовність 16 слів, тоді rowround (y) = (z_0, z_1, z_2,… z_{15})— також послідовність 16 слів,

де $(z_0, z_1, z_2, z_3) = quarterround(y_0, y_1, y_2, y_3)$,

$(z_5, z_6, z_7, z_4) = quarterround(y_5, y_6, y_7, y_4)$,

$(z_{10}, z_{11}, z_8, z_9) = quarterround(y_{10}, y_{11}, y_8, y_9)$,

$$(z_{15}, z_{12}, z_{13}, z_{14}) = quarterround(y_{15}, y_{12}, y_{13}, y_{14}).$$

columnround (y)

Тут ми беремо стовпці такий же матриці. Перетворимо їх функцією quarterround(y), по аналогії підставляючи в неї значення, починаючи з *j*-го для *j*-го стовпчика, де *j* = {0,1,2,3}.

Функція columnround (y) = (z) теж оперує послідовністю 16 слів так, що

$$(z_0, z_4, z_8, z_{12}) = quarterround(y_0, y_4, y_8, y_{12}),$$
$$(z_5, z_9, z_{13}, z_1) = quarterround(y_5, y_9, y_{13}, y_1),$$
$$(z_{10}, z_{14}, z_2, z_6) = quarterround(y_{10}, y_{14}, y_2, y_6),$$
$$(z_{15}, z_3, z_7, z_{11}) = quarterround(y_{15}, y_3, y_7, y_{11}).$$

doubleround (y)

Функція doubleround(y) є послідовним застосуванням функцій columnround, а потім rowround: doubleround (y) = rowround (columnround (y)).

$Salsa20(x) = x \oplus doubleround^{10}(x)$, де

$x = (x[0], x[1], \ldots, x[63]),$

$x_0 = littleendian(x[0], x[1], x[2], x[3]),$

$x_1 = littleendian(x[4], x[5], x[6], x[7]),$

…

$x_{15} = littleendian(x[60], x[61], x[62], x[63]).$

x[i] — байти *x*, а x_j — слова, які використовуються в описаних вище функціях.

Якщо , тоді Salsa20 (x) є послідовністю результатів

$$(z_0, z_1, \ldots, z_{15}) = doubleround^{10}(x_0, x_1, \ldots, x_{15})$$
$$littleendian^{-1}(z_0 + x_0), \ldots, littleendian^{-1}(z_{15} + x_{15})$$

Завдяки тому, що перетворення кожного стовпця і кожного рядка не залежать одне від одного, обчислення, необхідні для шифрування, легко распараллелівать. Це дає суттєвий виграш у швидкості для більшості сучасних платформ.

Алгоритм практично немає накладних обчислень, необхідних для запуску циклу шифрування. Це так само відноситься до зміни ключа. Багато шифросистемамі розраховують на попередні обчислення, результати яких повинні зберігатися в кеші першого рівня (L1) процесора. Так як вони залежать від ключа, обчислен-

ня доведеться проводити заново. У Salsa20 ж досить просто завантажити ключ у пам'ять.

Існує варіант алгоритму Salsa20, також запропонований Даніелем Бернштейном, в якому довжина nonce збільшена з 64 до 192 біт. Даний варіант називається XSalsa20. Збільшений розмір nonce дозволяє використовувати для його генерації криптографически стійкий генератор псевдовипадкових чисел, у той час як для безпечного шифрування з 64-бітовим nonce необхідне використання лічильника через високу ймовірність колізій.

Висновки: Використаний алгоритм Salsa має ряд переваг. По-перше, алгоритм Salsa є асиметричним, тобто він ґрунтується на поширенні відкритих ключів у мережі. Це дозволяє кільком користувачам обмінюватися інформацією, що посилається по незахищених каналах зв'язку. Також користувач сам може змінювати як числа, так і відкритий і закритий ключ на свій розсуд, тільки потім він повинен поширити відкритий ключ у мережі. Це дозволяє користувачеві отримати потрібну йому криптостійкість.

При всіх цих перевагах даний алгоритм має істотний недолік — невисока швидкість роботи. Алгоритм Salsa працює більш ніж у тисячу разів повільніше симетричного алгоритму DES.

З усього сказаного можна зробити висновок, що даний алгоритм шифрування, хоча досить повільний, але він асиметричний і дозволяє домагатися потрібної криптостойкости, що робить його незамінним при роботі в незахищених каналах зв'язку.

Література

1. Ященко В. В. Основные понятия криптографии // Математическое просвещение. Сер. 3. №2. 1998. С. 53-70.
2. https://en.wikipedia.org/wiki/Adiantum_(cipher)
3. https://en.wikipedia.org/wiki/Salsa20#ChaCha_variant
4. Кнут Д. Искусство программирования на ЭВМ. Т.2: Получисленные алгоритмы. М.: Мир. 1977.
5. Ахо А. Хопкрофт Дж. Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир. 1979.
6. Варновский Н. П. Криптография и теория сложности // Математическое просвещение. Сер. 3. №2. 1998. С. 71-86.
7. Василенко О. Н. Современные способы проверки простоты чисел // Кибернетический сборник, вып. 25. 1988. С. 162-188.
8. Прахар К. Распределение простых чисел. М.: Мир. 1967.
9.Боревич З.И. Шафаревич И.Р. Теория чисел. М.: Наука. 1964.
10. Кострикин А.И. Введение в алгебру. М.: Наука. 1977.
11. Брассар Дж. Современная криптология. Мир ПК. №3. 1997.

References

1. VV Yashchenko, Basic Concepts of Cryptography, Mathematical Enlightenment. Avg. 3. №2. 1998. S. 53-70.
2. https://en.wikipedia.org/wiki/Adiantum_(cipher)
3. https://en.wikipedia.org/wiki/Salsa20#ChaCha_variant
4. Knut D. The Art of Computer Programming. Vol.2: Semi-numerical algorithms. M .: Peace. 1977.
5. Aho A .. Hopcroft J .. Ullman J. Construction and analysis of computational algorithms. M .: Peace. 1979.
6. NP Barnovsky, Cryptography and the Theory of Complexity, Mathematical Enlightenment. Avg. 3. №2. 1998. pp. 71-86.
7. ON Vasilenko, Modern Methods for Checking the Simplicity of Numbers, // Cybernetic Collection, vol. 25. 1988, pp. 162-188.
8. Prahar K. Distribution of prime numbers. M .: Peace. 1967.
9. Z. Borevich Shafarevich IR Number theory. M .: Science. 1964
10. Kostrikin AI Introduction to Algebra. M .: Science. 1977.
11. J. Brasseur Modern cryptology. PC World. №3. 1997.

Статтю подано до редакції 31.01.2019 р.

УДК 339.727.22(045) DOI: 10.33111/mise.97.10

Гращенко І. С., к.е.н., доцент кафедри менеджменту зовнішньоекономічної діяльності підприємств, Національний авіаційний університет
Краснюк М. Т., к.е.н., доцент кафедри інформаційних систем в економіці, Київський національний економічний університет імені Вадима Гетьмана
Краснюк С. О., старший викладач кафедри іноземних мов, Київський національний університет технологій та дизайну

Hrashchenko I. S., PhD of Economics, Associate Professor, Department of Foreign Economic Activity Enterprise Management, National Aviation University
Krasnyuk M. T., PhD of Economics, Associate Professor of the Economics Information Systems Department, Kyiv National Economic University named after Vadym Hetman
Krasniuk S. A., Senior Lecturer, Department of Foreign Languages, Kyiv National University of Technologies and Design

УДОСКОНАЛЕННЯ ПРОЦЕСІВ ФОРМУВАННЯ ТА РЕАЛІЗАЦІЇ ІНВЕСТИЦІЙНОГО ПОТЕНЦІАЛУ АВІАПІДПРИЄМСТВ

IMPROVEMENT OF PROCESSES OF FORMATION AND REALIZATION OF INVESTMENT POTENTIAL OF AVIATION ENTERPRISES