

АТАКА ЗБОЇВ НА ARX-КРИПТОСИСТЕМУ NIGHT

С. Ю. БЛИНОВ^{1, a}

¹Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»,
Фізико-технічний інститут

Анотація

У даній роботі було запропоновано байтову атаку збоїв на шифр NIGHT та перевірено її ефективність. Використовуючи лише 4 фіксовані випадкові збої, вдалося зламати ключі останнього раунду шифрування, обираючи їх з 16 можливих.

Ключові слова: атаки збоїв, шифр NIGHT, ARX-криптосистеми

Вступ

В даному дослідженні ми розглянемо атаки збоїв (fault attacks) на шифри, побудовані на базі схеми Фейстеля, зокрема на ARX-шифр (від англ. Add-Rotation-Xor) NIGHT, що використовується для потреб «легкої» криптографії у Кореї.

ARX-криптосистеми як клас криптопримітивів сформувався у 2011 році і з того часу активно розвивається, оскільки побудовані лише із застосуванням операцій модульного та побітового додавання, а також циклічного зсуву бітових векторів. Ці операції є базовими для усіх сучасних процесорів, що дозволяє будувати надшвидкі алгоритми шифрування. Серед ARX-алгоритмів існують потокові шифри (Salsa, ChaCha), блокові шифри (Simon, Speck), геш-функції (BLAKE, Skein), що були досліджені з різних сторін, проте проблема стійкості до атак збоїв все ще залишається актуальною, що дає криптоаналітику з спеціальною технікою широкі можливості для злому і відновлення ключів.

Основні принципи, що використовуються зводяться до так званих диференціальних атак збоїв (differential fault attacks), оскільки аналітик використовує для відновлення ключа шифрування як коректні шифртексти, так і шифртексти, одержані після втручання у процес шифрування (збиті шифртексти); знання відкритого тексту, що шифрується, не вимагається. Ці атаки використовують апарат математичної статистики; фактично на одержаному під час аналізу матеріалі будуються розпізнавачі для некоректних ключів.

1. Теоретичні відомості

Загальна структура шифру NIGHT

Ми використовуємо такі позначення для опису NIGHT. 64-бітний текст і шифр-текст розглядаються як об'єднання з 8 байтів і позначаються відповідно $P = P_0 || \dots || P_6 || P_7$ і $C = C_0 || \dots || C_6 || C_7$. 64-

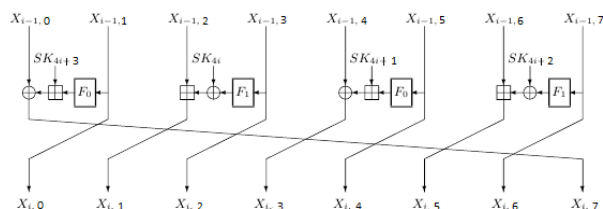


Рис. 1. Схема раундової функції, де $i = \overline{1, 31}$

бітні проміжні значення представлені аналогічно: $X_i = X_{i,0} || \dots || X_{i,6} || X_{i,7}$ для $i = \overline{0, 32}$. 128-бітний ключ (masterKey) розглядається як об'єднання 16 байтів і позначається $MK = MK_0 || \dots || MK_{15}$. Відповідно з МК за допомогою процедури keySchedule отримуємо набір $WK = WK_0 || \dots || WK_6 || WK_7$ (whiteningKey) і $SK = SK_0 || \dots || SK_{127}$ (subKey). Наявність блокової структури у раундовій функції (Рис. 1) дуже спрощує аналіз, оскільки для блокової структури можна перебирати раундовий ключ частинами, по кожній з яких є відносно невелика кількість варіантів. Зауважимо, що перетворення внутрішньої раундової функції для нас не важливі

Загальна схема атаки

Сутність атаки полягає у внесенні помилки в оброблений блок даних безпосередньо перед останнім раундом або за кілька раундів до останнього (у нашому випадку це 26-31 раунди) та порівнянні одержаного «збитого» шифртексту із оригінальним за допомогою функцій розпізнавання, що в нашому випадку будуються в залежності від місця проведення збою:

$$g(WK, SK) = (\tilde{C}_i \oplus (f_0(\tilde{C}_{i+1} \oplus WK) + SK)) \oplus$$

$$(C_i \oplus (f_0(C_{i+1} \oplus WK) + SK)), i = 0, 4$$

$$g(WK, SK) = (\tilde{C}_i - (f_1(\tilde{C}_{i+1} - WK) \oplus SK)) \oplus$$

$$(C_i - (f_1(C_{i+1} - WK) \oplus SK)), i = 2, 6$$

^ablinoffhouse@ukr.net

де \tilde{C}_i «збитий» шифртекст
 C_i – оригінальний шифртекст

Ми розглядаємо байтову випадкову модель збоїв, тобто змінюємо значення одного з восьми байтів на випадкове.

Через блокову структуру шифру маємо дві моделі розповсюдження помилки:

- 1) збій в $X_{i,0}, X_{i,2}, X_{i,4}, X_{i,6}$ байтах
- 2) збій в $X_{i,1}, X_{i,3}, X_{i,5}, X_{i,7}$ байтах

Алгоритм атаки виглядає наступним чином:

1) Згенерувати N шифртекстів та виконати N збоїв (Випадковий збій в кожному).

2) Для кожної припустимої пари ключів порахувати розподіл значень функцій розпізнавання $g^{(j)}$.

3) Перевірити гіпотезу « $g^{(j)}$ має такий самий розподіл як помилка, що вноситься до шифртекстів».

Розрізняти гіпотези будемо за допомогою Евклідової відстані за формулою:

$$d(WK, SK) = \sum_{x=0}^{256} \left(\frac{\#(g^{(j)}(WK, SK) = x)}{N} - \frac{1}{256} \right)^2$$

де $\#(g^{(j)}(WK, SK) = x)$ означає кількість значень по всіх шифртекстах при фіксованих WK, SK . Розподіл значень припускаючих функцій $g^{(j)}$ залежить від розподілу внесених у 1000 шифртекстів помилок і очікується близьким до рівномірного, якщо помилки у $X_{i,0}, X_{i,2}, X_{i,4}, X_{i,6}$ байтах немає, і нерівномірним, якщо є.

2. Експериментальна перевірка

Було побудовано модель шифру HIGHT на мові програмування Java, використовуючи функцію SecureRandom для генерування випадкової помилки. Далі для кожної з 65536 пар ключів обчислюємо значення $d(WK, SK)$ для 1000 шифртекстів (звичайних та збитих).

Оскільки функція SecureRandom дає статистично рівномірний розподіл помилки, знаходити *максимальну* евклідову відстань від рівномірного розподілу виявилось неефективно, тому було прийнято рішення вносити помилки, що не впливають на $X_{i,0}, X_{i,2}, X_{i,4}, X_{i,6}$ байти і аналізувати випадки, коли перед останнім раундом збитими виявляються $X_{i,1}, X_{i,3}, X_{i,5}, X_{i,7}$ байти, причому збиті байти не повинні накладатися. За таких умов правильна пара ключів буде відповідати конкретному значенню відстані

$$d(WK, SK) = 0.0046544912109375,$$

що було визначено експериментальним чином.

Таких пар (WK, SK) буде в середньому 8, тож один з восьми WK та відповідний йому SK при одному збої буде знайдено з ймовірністю $1/8$. Відповідно для відновлення всіх 4 байтів SK та 4 байтів WK потрібно 4 випадкові збої у $X_{i,1}, X_{i,3}, X_{i,5}, X_{i,7}$ байтах останнього раунду та в будь-якому байті з 26 по

30 раунд. Оскільки раундові ключі відрізняються від МК на відомому константу, то знаходження раундових ключів дозволяє знаходити МК. Відновивши ключі останнього раунду, атака повторюється для попереднього, поки весь МК не буде знайдено.

3. Висновки

У даній роботі було перевірено стійкість шифру HIGHT до атак збоїв, використовуючи байтові випадкові помилки у останніх 6 раундах. Обрана модель є не дуже жорсткою для криптоаналітика, проте дозволяє досить швидко знаходити можливі варіанти для ключів, що суттєво зменшує перебір. Також було визначено ефективну стратегію внесення помилок. Для того, щоб покращити алгоритм знаходження ключів, криптоаналітику потрібно вносити не випадкові, а обрані помилки, що дозволить контролювати розподіл значень $g^{(j)}$ і користуватися максимальною Евклідовою відстанню.

Для визначення інших раундових ключів та відновлення вихідного ключа шифрування (або невідомого відкритого тексту) потрібно проводити атаку ще таку кількість раз, скільки раундів має перетворення.

Захист реалізації шифрів від такого роду атак полягає у дублюванні останніх раундів шифрування: дані обробляються в незалежних контурах, після чого йде звірка результатів, проте для повного захисту потрібно дублювати всі раунди, що небажано для легких шифрів.

Перелік використаних джерел

1. Eli Biham, Adi Shamir, "Differential fault analysis of secret key cryptosystems" – LNCS 1294/1997 – pp. 513-525
2. Matthieu Rivain, "Differential Fault Analysis on DES Middle Rounds" – LNCS 5747/2009 – pp. 457-469
3. Deukjo Hong, Jaechul Sung, Seokhie Hong, "HIGHT: A New Block Cipher Suitable for Low-Resource Device" – Center for Information Security Technologies (CIST), Korea University, Seoul, Korea – 2006
4. D. Naccache, "Finding faults [data security]," – IEEE Security and Privacy – vol. 3, no. 5, pp. 61-65, 2005.
5. Dilip Kumar, Sikhar Patranabis, Jakub Breier, "A Practical Fault Attack on ARX-like Ciphers with a Case Study on ChaCha20" – SEAL, Department of Computer Science and Engineering, IIT Kharagpur, India – 2017
6. J. Takahashi, T. Fukunaga, and K. Yamakoshi., DFA Mechanism on the AES Key Schedule. In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, Fault Diagnosis and Tolerance in Cryptography – FDTC 2007, pages 62-74. – IEEE Computer Society, 2007.