Комп'ютерний практикум 2

Статистична атака на комбінувальний генератор гами

Мета роботи

Практична реалізація статистичної атаки на комбінувальний генератор гами; набуття навичок роботи з системами комп'ютерної алгебри.

Необхідні теоретичні відомості

Застосування комбінувальних функцій є поширеним методом побудови генераторів гами. Нагадаємо, що комбінувальний генератор гами складається з таких компонентів (рис. 1).

1. Набір регістрів зсуву $R_1, \ldots, R_n$ з поліномами зворотного зв'язку $p_1(x), \ldots, p_n(x)$. Будемо вважати, що довжина всіх регістрів однакова і дорівнює m. Кожен поліном зворотного зв'язку виду $p(x) = x^m \oplus c_{m-1}x_{m-1} \oplus \ldots \oplus c_1 x_1 \oplus c_0$ задає рекурентну послідовність виду $x_l = c_{m-1}x_{l-1} \oplus \ldots \oplus c_0 x_{l-m}$, $l \geq m$.

2. Комбінувальна функція $f : V_n \to \{0, 1\}$. Зазначимо, що в i-ому такті біт гами визначається за формулою
$$\gamma_i = f(x_i^{(1)}, \ldots, x_i^{(n)}), \ i = 1, 2, \ldots.$$

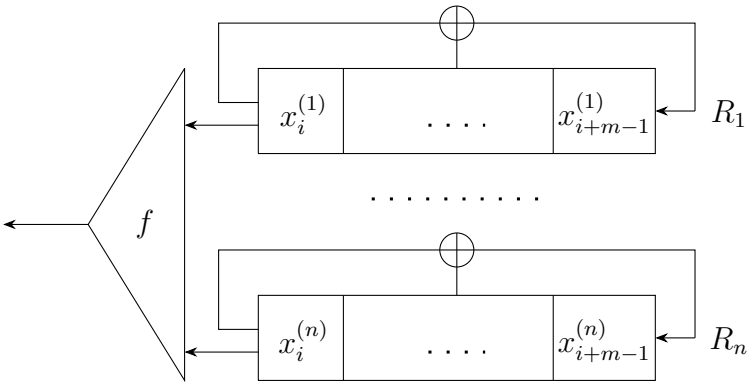

Рис. 1: Фільтрувальний генератор гами

Припустимо, що відомо N знаків гами $\gamma_1, \ldots, \gamma_N$. Розглянемо статистичну атаку на комбінувальний генератор гами, яка вперше була запропонована Зігенталером. Ця атака використовує декілька припущень: по-перше, всі знаки всіх початкових станів генеруються незалежно рівноймовірно, по-друге, існує функція $g : V_k \to \{0, 1\}$ від меншої кількості змінних $k \leq n$ така, що

$$\Pr\left(f(X_1, \ldots, X_n) \neq g(X_{i_1}, \ldots, X_{i_k})\right) = p \leq 1/2 \cdot (1 - \varepsilon), \ \varepsilon > 0,$$

де $1 \leq i_1 \leq \ldots \leq i_k \leq n$, $(X_1, \ldots, X_n)$ – випадковий рівноймовірний вектор з V_n.

Інакше кажучи, для функції f існує апроксимація g, яка залежить від меншої кількості змінних, тобто початковий генератор гами має більш простий генератор-аналог, який в якості комбінувальної функції використовує g. Якщо така функція g знайдена, то це надає змогу

відновити початкові стани регістрів з номерами $i_1, \ldots, i_k$. Для цього можна скористатись методом максимуму правдоподібності. Напочатку потрібно зафіксувати значення $T \leq N$, що визначає кількість матеріалу, необхідну для надійного відновлення початкових станів регістрів зсуву $R_{i_1}, \ldots, R_{i_k}$. Далі потрібно виконати такі дії.

1. Перебрати початкові стани регістрів зсуву $R_{i_1}, \ldots, R_{i_k}$. Для кожного набору початкових станів обчислити відрізок гами довжини T, згенерований генератором гами з комбінувальною функцією g. Для кожного відрізку гами обчислити статистику, яка дорівнює кількості розбіжностей між цим згенерованим відрізком гами та $\gamma_1, \ldots, \gamma_T$.

2. Серед усіх наборів початкових станів обрати той набір, що має найменше значення статистики, тобто згенерований цим набором відрізок гами має найменшу кількість розбіжностей з $\gamma_1, \ldots, \gamma_T$ (відстань Гемінга до $\gamma_1, \ldots, \gamma_T$).

3. Знайти значення інших регістрів зсуву повним перебором. Зауважимо, що якщо функція f має декілька апроксимацій, то можна застосовувати атаку декілька раз ітеративно (не обов'язково шукати інші початкові стани регістрів суто повним перебором).

Зрозуміло, що якщо виконуються припущення атаки, то значення статистики для істинного набору початкових станів буде дорівнювати приблизно Tp, в той час як як для інших хибних наборів вона дорівнюватиме приблизно $T/2$. Наведемо результат, який надає змогу визначити потрібну кількість матеріалу T (в припущенні, що відстань єдиності вхідного генератора гами є величиною порядку nm).

Твердження 1. *Нехай виконуються припущення атаки,* $\delta \in (0,1)$ *і*

$$T = \lceil 8 \cdot \varepsilon^{-2} \ln \left(2^{mk} \delta^{-1}\right) \rceil.$$

Тоді описана вище атака відновлює початковий стан комбінувальноного генератора гами з ймовірністю не менше $1 - \delta$ *та часовою складністю*

$$O\left(T \cdot 2^{mk} + nm \cdot 2^{m(n-k)}\right).$$

Залишається зауважити, що наближення функції f від меншої кількості кількості змінних можна шукати серед афінних статистичних аналогів f. Для цього достатньо обчислити коефцієнти Уолша-Адамара $\hat{f}(\alpha)$, $\alpha = (\alpha_1, \ldots, \alpha_n) \in V_n$, функції f та обрати ті значення серед них, що не дорівнюють нулю. Якщо для деякого α виконується $\hat{f}(\alpha) < 0$, то отримуємо статистичний аналог $\alpha_1 X_1 \oplus \ldots \oplus \alpha_n X_n \oplus 1$, якщо $\hat{f}(\alpha) > 0$, то аналогом буде $\alpha_1 X_1 \oplus \ldots \oplus \alpha_n X_n$. Окрім цього, коефіцієнт Уолша-Адамара одразу ж надає змогу оцінити якість такого статистичного аналогу: якщо $c \in \{0,1\}$, $(X_1, \ldots, X_n)$ – випадковий рівноймовірний вектор з V_n, то виконується рівність

$$\Pr\left(f(X_1, \ldots, X_n) = \alpha_1 X_1 \oplus \ldots \alpha_n X_n \oplus c\right) = 1/2 \cdot (1 + (-1)^c 2^{-n} \hat{f}(\alpha)).$$

Зазначимо, що стійкість булевих функцій по відношення до атак такого типу визначає параметр, який називається *кореляційною імунністю*: функція f є кореляційно-імунною порядку k, $k \in \overline{1, n-1}$, якщо для будь-якого набору $1 \leq i_1 \leq \ldots \leq i_k \leq n$ випадкові величини $f(X_1, \ldots, X_n)$ та $(X_{i_1}, \ldots, X_{i_k})$ є незалежними. Цей параметр може бути обчислений за допомогою коефіцієнтів Уолша-Адамара: функція f буде кореляційно-імунною порядку k, якщо $\hat{f}(\alpha) = 0$ для кожного ненульового вектора $\alpha \in V_n$, що має вагу не вище k. Через $\mathrm{cor}\,(f)$ позначають найбільше число k, для якого функція f є кореляційно-імунною порядку k; якщо такого k не існує, то вважають, що $\mathrm{cor}\,(f) = 0$. Про кореляційну імунність та про атаку Зігенталера див. детальніше в навчальному посібнику.

Система комп'ютерної алгебри SAGE

Система комп'ютерної алгебри SAGE містить функцію lfsr_sequence для генерації послідовностей лінійних регістрів зсуву, яка приймає на вхід довільний поліном зворотного зв'язку та початковий стан і повертає необхідну кількість знаків. Однак, ця функція не є оптимальною для великих обчислень, тому для оптимізації можна використовувати Cython, який підтримується в SAGE. Cython, фактично, є компільованою мовою програмування, що базується на Python, і надає змогу транслювати код на Python в код на C. Це, в свою чергу, надає змогу в декілька разів прискорити певні функції.

Розглянемо приклад аналогічної до lfsr_sequence функції на Cython для регістру зсуву довжини 10 (щоб писати код на Cython в Jupyter Notebook достатньо додати в комірку «магічну команду» %%cython). Зауважимо, що в цьому прикладі задля оптимізації в якості типу даних для зберігання стану регістра використовується int (який транслюється в 32-бітовий int в C).

```
%%cython
from libc.stdlib cimport malloc, free
cdef int LENGTH = 10

def gen_sequence(int poly, int init_state, int nbits):
    cdef int curr_state = init_state, s, i, j
    cdef int *res_array = <int *>malloc(sizeof(int) * nbits)
    try:
        for i in range(nbits):
            res_array[i] = (curr_state >> (LENGTH - 1)) & 1
            s = 0
            for j in range(LENGTH):
                s = s ^ (((curr_state & poly) >> j) & 1)
            curr_state = (curr_state << 1) | s
        return [x for x in res_array[:nbits]]
    finally:
        free(res_array)
```

Порівняємо час роботи функцій lfsr_sequence та gen_sequence:

```
import time
init_state = [1, 0, 0, 1, 0, 0, 0, 1, 1, 1]
poly = x^10 + x^3 + 1
st_time = time.time()
seq1 = lfsr_sequence(list(vector(GF(2), poly.list()[:-1])),
                     list(vector(GF(2), init_state)), 1 << 24)
end_time = time.time()
print('SAGE function time: {0}'.format(end_time - st_time))

init_state_int = int(''.join([str(i) for i in init_state]), 2)
poly_int = int(''.join([str(i) for i in poly.list()[:-1]]), 2)
st_time = time.time()
seq2 = gen_sequence(poly_int, init_state_int,  1 << 24)
end_time = time.time()
print('Cython function time: {0}'.format(end_time - st_time))

print(seq1 == seq2)
>>SAGE function time: 185.05702471733093
>>Cython function time: 1.8951385021209717
>>True
```

Таким чином, функція gen_sequence є еквівалентною lfsr_sequence, але надає перевагу в приблизно 100 разів в часі виконання. Подобного типу оптимізації можна використовувати для імплементації методу максимуму правдоподібності та інших допоміжних функцій в даному комп'ютерному практикумі.

Вхідні дані

1. Шість лінійних регістрів зсуву довжини 10 з поліномами зворотного зв'язку

$$p_1(x) = x^{10} \oplus x^3 \oplus 1,\ p_2(x) = x^{10} \oplus x^7 \oplus 1,$$
$$p_3(x) = x^{10} \oplus x^4 \oplus x^3 \oplus x^1 \oplus 1,\ p_4(x) = x^{10} \oplus x^8 \oplus x^3 \oplus x^2 \oplus 1,$$
$$p_5(x) = x^{10} \oplus x^8 \oplus x^4 \oplus x^3 \oplus 1,\ p_6(x) = x^{10} \oplus x^8 \oplus x^5 \oplus x^1 \oplus 1.$$

2. Комбінувальні функції $f(x_0, \ldots, x_5)$ генераторів гами для всіх варіантів містяться в архіві під назвою `Variants_CombFunction.rar`.

3. Відрізки гами $\gamma_1, \ldots, \gamma_N$ довжини 50000 бітів для всіх варіантів містяться в архіві під назвою `Variants_Gamma.rar`.

Порядок виконання роботи

1. Обчислити коефіцієнти Уолша-Адамара функції f. За допомогою цих коефіцієнтів знайти всі афінні статистичні аналоги f та ймовірності збігу цих аналогів з f.

2. Вибрати набір статистичних аналогів $g_1, \ldots, g_r$, що будуть використані для відновлення початкового стану генератора. Для всіх функції з цього набору обчислити необхідну кількість матеріалу $T_1, \ldots, T_r$ відповідно, зафіксувавши $\delta = 0.01$. Побудувати низку атак на початові стани регістрів, тим самим повністю відновивши початковий стан генератора гами.

3. Перевірити, що початковий стан генерератор гами відновлено правильно, згенерувавши відрізок гами відповідної довжини й порівнявши його з вхідними даними.

Оформлення протоколу

Протокол до комп'ютерного практикуму оформлюється згідно зі стандартними правилами оформлення наукових робіт. До протоколу можна не включати анотацію, перелік термінів та позначень та перелік використаних джерел. Також не обов'язково оформлювати зміст.

Протокол має містити:

- мету комп'ютерного практикуму;
- постановку задачі, варіант завдання та хід роботи;
- всі афінні статистичні аналоги та ймовірності збігу з функцією f;
- значення $\mathrm{cor}\,(f)$;
- перелік статистичних аналогів $g_1, \ldots, g_r$, які було обрано; для кожної з цих функцій треба вказати необхідну кількість матеріалу $T_1, \ldots, T_r$, мінімальні значення статистик (абсолютні й відносні) при побудові відповідних атак, час виконання кожної атаки.
- знайдений початковий стан генератора гами;
- програмний код для знаходження початкового стану генератора гами;
- висновки.

Оцінювання комп'ютерного практикуму

Оцінка за комп'ютерний практикум складається з двох складових: оцінки за презентацію отриманих студентом результатів, яка складає 6 балів, та оцінки за протокол, яка також складає 6 балів.

Якщо знайдений в ході виконання комп'ютерного практикуму початковий стан є неправильним, то здача комп'ютерного практикуму не зараховується. В такому разі студенту надається можливість переробити комп'ютерний практикум.

Під час презентації отриманих результатів студент надає викладачу в усній формі звіт по виконаній роботі. В процесі цього викладач може ставити уточнювальні питання по програмному коду, а також теоретичні питання по матеріалах курсу, які стосуються цього комп'ютерного практикуму. Окрім цього, викладач може поставити питання до реалізації тих чи інших функцій у використаних студентом програмних реалізаціях. Якщо студент не орієнтується в програмному коді своєї роботи або не володіє відповідним теоретичним матеріалом, то відповідь не зараховується, і йому у майбутньому буде надано ще одну спробу для презентації. Дата та час повторної здачі врегульовується за домовленістю з викладачем в індивідуальному порядку.

Критерії оцінювання презентації отриманих результатів та протоколу містяться в положенні про рейтингову систему оцінювання.

Здача протоколу після назначеного терміну виконання без поважної причини приводить до зниження оцінки за нього на 1 бал за кожен тиждень запізнення (перший бал втрачається одразу після назначеного терміну виконання); при цьому оцінка не опускається нижче нуля. Тижнем в даному випадку вважається 7 календарних днів. Якщо в програмному коді та/або протоколі присутні ознаки плагіату, то студент отримує за нього нуль балів без можливості повторного виконання цього завдання.