Лекция 5. Метод «встреча по середине»

Цель лекции: рассмотреть один из универсальных методов криптоанализа.

План лекции:
Введение.
1 Метод «встреча по середине»
Заключение
Контрольные вопросы

Ключевые слова: [выбрать самостоятельно].

Содержание лекции:

Введение

В криптоанализе методом встречи посередине или атакой «встречи посередине» (англ. meet-in-the-middle attack) называется класс атак на криптографические алгоритмы, асимптотически уменьшающих время полного перебора за счет принципа «разделяй и властвуй», а также увеличения объема требуемой памяти.

1 Метод «встреча по середине»

Рассмотрим каскадный метод построения сложного шифра из исходных простых. Даны два шифра $T_1(x, k_1)$ и $T_2(z, k_2)$. Шифртекст y получается из открытого текста x композицией отображений T_1 и T_2, то есть

$$y = T_2(T_1(x, k_1), k_2).$$

Ключ шифра – вектор $k = (k_1, k_2)$, где $k_1 \in K_1$, $k_2 \in K_2$, $K = K_1 \times K_2$. Разумеется, в T_1 и T_2 согласованы области определения и значений при любых $k_1 \in K_1$ и $k_2 \in K_2$ и расшифрование происходит применением следующего преобразования:

$$x = T_1^{-1}(T_2^{-1}(y, k_2), k_1).$$

Трудоемкость полного перебора равна $|K_1||K_2|/2$. Вместе с тем можно сократить перебор, увеличив используемую память [1, с. 235]. Пусть для известной пары (x, y) ключ (k_1, k_2) определяется единственным образом. Для всех $k_1^{(i)} \in K_1$, $i = 1, ..., |K_1|$, построим таблицу

$$\begin{gathered} z_1 = T_1\left(x, k_1^{(1)}\right) \\ \dots\dots\dots\dots\dots\dots\dots \\ z_{|K_1|} = T_1\left(x, k_1^{|K_1|}\right). \end{gathered} \qquad (1)$$

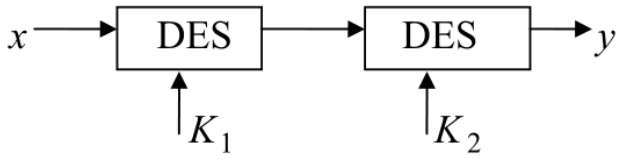
Вычисление таблицы (1) потребует $|K_1|$ операций опробования. Построим следующую таблицу для всех $k_2^{(j)} \in K_2,\ j = \overline{1, |K_2|}$.

$$\begin{gathered} z_1' = T_2^{-1}\left(y, k_2^{(1)}\right) \\ \dots\dots\dots\dots\dots\dots\dots \\ z_{|K_2|}' = T_2^{-1}\left(y, k_2^{(|K_2|)}\right) \end{gathered} \qquad (2)$$

[1] Menzes A., van Oorschot P., Vanstone S. Hadbook of Applied Cryptography, CRC Press, 1997.

Вычисление (2) потребует $|K_2|$ операций опробования. Объединим таблицы (1) и (2) и проведем их упорядочение в соответствии с некоторым порядком на множестве значений промежуточных шифртекстов z. Известно [2], что сложность упорядочения таблицы размера $(|K_1| + |K_2|)$ оценивается величиной $(|K_1| + |K_2|)\ln(|K_1| + |K_2|)$. Пара $z_i = z'_j$ при некоторых (i, j) определяет использованный ключ $(k_1^{(i)}, k_2^{(j)})$. Для ее нахождения достаточно просмотреть таблицу. Число операций, которые мы затратили $(|K_1| + |K_2|) + (|K_1| + |K_2|)\ln(|K_1| + |K_2|)$. Если $|K| = N$ и $|K_1| = |K_2|$, то метод «встреча по середине» дает оценку $\sqrt{N}\ln N$, что значительно меньше N при $N \to \infty$. Такой метод требует памяти размера 2N.

Пример 1. Двойной DES. Пусть для повышения стойкости используется повторное шифрование алгоритмом DES на разных ключах K_1 и K_2.



Тогда метод «встреча посередине» дает оценку трудоемкости $2^{56}\ln 2^{112} \approx 100 \cdot 10^{17} = 10^{19}$, что значительно меньше полного перебора ($\approx 10^{34}$). Однако требуемая память $\approx 10^{17}$ является значительной.

Пример 2. Рассмотрим следующий шифр, допускающий быструю реализацию на ЭВМ. Для произвольного целого неотрицательного числа n обозначим <n> – двоичное представление n, *l*<n> – длина двоичного представления n, [n] – разбиение двоичного числа n на байты (дописав слева нули, если необходимо). Для вектора целых неотрицательных чисел $\overline{n} = (n_1, n_2, ..., n_s)$ положим $<\overline{n}> = (<n_1>,<n_2>, ..., <n_s>)$, $[\overline{n}] = ([n_1], [n_2], ..., [n_s])$.

Рассмотрим симметричную группу подстановок S_{256} и для любого $g \in S_{256}$ будем отождествлять записи:

$$g = \begin{pmatrix} \dots m \dots \\ \dots i_m \dots \end{pmatrix} = \begin{pmatrix} \dots < m > \dots \\ \dots < i_m > \dots \end{pmatrix} = \begin{pmatrix} \dots [m] \dots \\ \dots [i_m] \dots \end{pmatrix}, \qquad m = 0, \dots, 255.$$

Пусть дана таблица $\mathfrak{J}$ подстановок из S_{256}: $g_0, ..., g_{M-1}$, где M = 1024. Эта таблица записана в памяти так, что для i = 0, ..., M – 1, подстановка g_i имеет адрес <i> длины *l*<i> = 10. Тогда адреса $a = (k_1, ..., k_8)$ 8 подстановок задаются двоичным вектором $<a> = (<k_1>, ..., <k_8>)$, $1<a> = 80$. С другой стороны $[a] = ([r_1], ..., [r_8])$ – вектор из 10 байт. Пусть открытый текст x представим в виде последовательности байт $x = (x_1, ..., x_l)$. В качестве шифра рассмотрим простую замену байт открытого текста по подстановке $g \in S_{256}$, полученной по правилу:

$$g = g_{k_1} \cdots g_{k_8},$$

где $<\overline{k}> = (<k_1>, ..., <k_8>)$ – 80-битный ключ шифрования, который выбирается случайно и равновероятно, $k_i = 0, ..., M - 1$, $l<k_i> = 10$, g_{k_i} взято из таблицы $\mathfrak{J}$. Пусть даны открытый текст x и шифртекст y. Рассмотрим атаку «встреча по середине» при условии, что таблица подстановок известна противнику. Разобьем ключ на две 40-битных половины и будем опробовать каждую из них, строя две таблицы:

---

[2] Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980.

$$z^{(k_1k_2k_3k_4)} = g_{k_1}g_{k_2}g_{k_3}g_{k_4}(x); \quad k_1, k_2, k_3, k_4 = 0, \dots, M-1,$$
$$\tilde{z}^{(k_5k_6k_7k_8)} = g_{k_8}^{-1}g_{k_7}^{-1}g_{k_6}^{-1}g_{k_5}^{-1}(y); \quad k_5, k_6, k_7, k_8 = 0, \dots, M-1.$$

Эти вычисления потребуют $2(1024)^4 = 2^{41} \approx 2 \cdot 10^{12}$ операций опробования и памяти $\approx 2^{41} \times l$ байт.

Проведем единое упорядочивание двух таблиц и выбор пар одинаковых последовательностей z. Сложность упорядочивания оценивается следующим образом

$$2^{41} \cdot l \cdot \ln(2^{41} l) \approx 41 \cdot 2^{41} \cdot l \cdot \ln 2 \approx 8 \cdot l \cdot 10^{13}$$

операций.

Заключение

В некоторых случаях бывает сложно разделить биты последовательности ключей на части, относящиеся к разным ключам. В таком случае применяют алгоритм 3-subset MITM attack предложенный Богдановым и Ричбергером в 2011 году на основе обычного метода встречи посередине.

Контрольные вопросы

Смотри руководство по организации самостоятельной работы магистрантов.