А. Г. Ростовцев

Алгоритм Полларда невозможно улучшить за счет увеличения объема памяти

1. Введение

Многие задачи криптоанализа, например, логарифмирование в группе вычислимого порядка или поиск коллизий хэш-функции, сводятся к задаче о встрече на графе случайного отображения, которая решается алгоритмом Полларда [1]. Этот алгоритм использует сжимающие свойства случайного отображения, граф которого представляет собой направленный лес, а корни деревьев связаны в циклы. Алгоритм Полларда для логарифмирования в группе порядка r имеет сложность $O(\sqrt{r})$, требуемый объем памяти равен $O(\log r)$; этот алгоритм не поддается эффективному распараллеливанию.

Другой популярный метод логарифмирования и поиска коллизий — алгоритм встречи посередине [2] — имеет сложность $O(\sqrt{r}\log r)$, требует такого же объема памяти, но при этом может быть распараллелен. При параллельной работе n процессоров, где $n << \sqrt{r}$, время работы алгоритма уменьшается в n раз. Этот алгоритм не использует сжимающие свойства случайного отображения.

Промежуточным вариантом между алгоритмами Полларда и встречи посередине является алгоритм встречи на случайном дереве. В этом случае для случайных стартовых вершин делается несколько шагов по графу случайного отображения, конечные вершины запоминаются и среди них ищутся равные. Если число шагов равно $O(\log r)$, то сложность алгоритма не может превышать $O(\sqrt{r\log r})$ [3]. В работе [4] рассматривается распараллеливание алгоритма о встрече, однако сжимающие свойства случайного отображения не учтены. Теоретическая оценка сложности алгоритма о встрече из литературы неизвестна.

2. Строение случайного дерева

Приведем основные свойства графа случайного отображения. Известно [5], что если число вершин велико, то почти все графы имеют одинаковые статистические свойства. Почти все вершины лежат на одном дереве. Высота дерева и длина цикла равны $O(\sqrt{r})$. При обращении стрелок случайное дерево описывается критическим ветвящимся процессом [5]. Каждая вершина может быть представлена как частица, которая живет одно поколение и дает число потомков распределенное по закону Пуассона с параметром $\lambda = -1$.

Определим глубину вершины дерева как максимальное расстояние от нее до какого-либо из листьев. Доля вершин с глубиной 0 равна exp(–1), доля вершин с глубиной не более 1 равна exp(–1 + exp(–1)), ..., доля вершин с глубиной d равна $\underbrace{\exp(-1+\exp(-1+...+\exp(-1)...))}_{d \text{ раз}}$. При больших глубинах доля вершин с глубиной более d, асимптотически равна $\frac{2}{d}$, доля вершин с глубиной ровно d асимптотически равна $\frac{2}{d^2}$ (в действительности эта величина всегда меньше чем $\frac{2}{d^2}$).

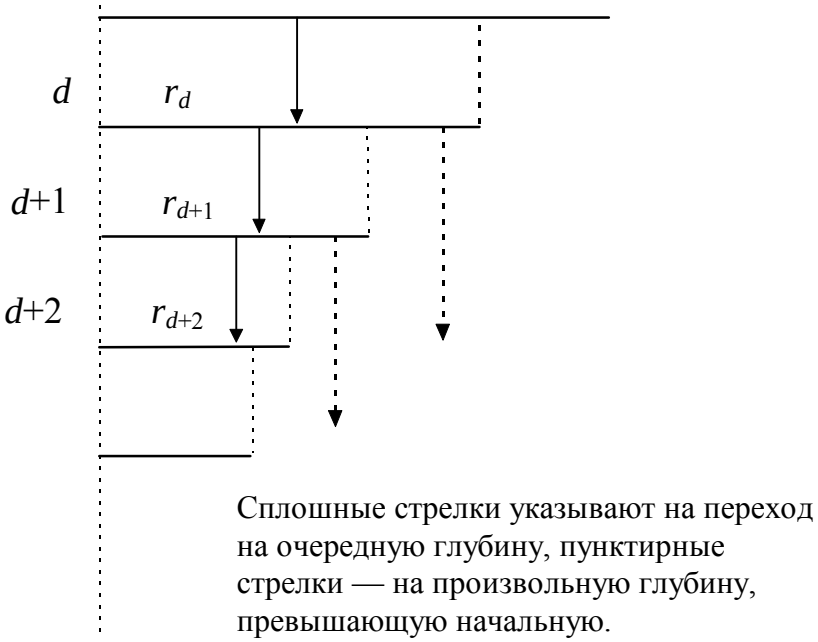

Рис. 1. Строение случайного дерева

Обозначим через $P(d)$ вероятность того, что случайная вершина имеет глубину более d; $Q(d) = 1 - P(d)$; $p(d)$ — вероятность того, что вершина имеет глубину d; $p_k(d)$ вероятность попасть на глубину d после k шагов при старте от листа, $p_1(d_2|d_1)$ — вероятность попасть после одного шага с глубины d_1 на глубину d_2. При этом выполняется равенство $Q(d+1) = e^{-1+Q(d)}$.

Каждая из r_d вершин с глубиной $d > 0$ связана с вершиной с глубиной $d - 1$ (и, возможно, с другими вершинами с глубиной не более $d - 1$), как показано на рисунке. Доля вершин, которые переходят не на очередную глубину, а имеют пропуски, равна $\sum_i p(i-1) - p(i) \approx Q(0) = \frac{1}{e}$.

3. Алгоритм встречи на случайном дереве

Для встречи на случайном дереве будем делать k шагов, конечные вершины сортировать и искать среди них равные. После k шагов встреча возможна только на глубинах не менее k. Пусть $k = O(\log r)$ и предположим, что встреча равновероятна в любой вершине с глубиной не менее k. Число вершин, в которых возможна встреча, равно $O\left(\frac{r}{\log r}\right)$. В соответствии с парадоксом дней рождения, получаем сложность этапа создания базы данных $O\left(\sqrt{\frac{r}{\log r}}\right)$, а с учетом сортировки базы данных сложность алгоритма составит $O(\sqrt{r \log r})$. На самом деле предположение неверно и встреча не равновероятна в любой вершине. Это обстоятельство дополнительно повышает вероятности встречи и снижает сложность алгоритма.

Запрет перехода вершины с глубиной d_1 на ту же или меньшую глубину можно учесть с помощью условной вероятности. Для этого достаточно положить, что вершина может переходить на любую глубину d_2, превышающую заданную глубину d_1, т.е. вероятности задаются дробью $\frac{p(d_2)}{P(d_1)}$. С учетом того, что каждая вершина глубины $d_1 + 1$ соединена с вершиной глубины d_1, получаем следующую формулу для вероятности перехода за один шаг:

$$p_1(d_2 | d_1) = \begin{cases} \dfrac{p(d_1+1)}{p(d_1)}, & \text{если} \quad d_2 = d_1 + 1 \\ \dfrac{p(d_2)(p(d_1) - p(d_1+1))}{P(d_1)}, & \text{если} \quad d_2 \geq d_1 + 1. \end{cases} \qquad (1)$$

Вероятность перехода на очередную глубину с учетом частично перекрывающихся условий (1) можно записать следующим образом:

$$p_1(d_1{+}1|d_1) = p(d_1+1)\left(\frac{1}{p(d_1)} + P(d_1)\right).$$

Заменяя $p(d_1) \approx \frac{2}{{d_1}^2}$ и $P(d_1) \approx \frac{2}{d_1 + 1}$, получим $p_1(d_1{+}1|d_1) \approx 1 - Q(d+1)$.

С учетом того, что $\frac{p(d_1) - p(d_1+1)}{P(d_1)} \approx \frac{2}{d_1}\frac{d_1}{2} = 1$, можно записать (1) в виде

$$p_1(d_2 | d_1) = \begin{cases} Q(d_1+1), & \text{если} \quad d_2 = d_1 + 1 \\ p(d_2), & \text{если} \quad d_2 > d_1 + 1. \end{cases} \qquad (2)$$

Из (2) следует, что в случае больших глубин вероятность пропуска очередной глубины при одиночном шаге падает пропорционально квадрату глубины, а вероятность попасть на глубину d_2 в результате такого пропуска приблизительно равна безусловной вероятности попасть на глубину d_2, т.е. не зависит от d_1. Процесс спуска по случайному дереву является марковским с матрицей $L = (p_{ij})$ переходных вероятностей $p_1(i|j) = p_{ji}$:

$$L = \begin{pmatrix} 0 & Q(1) & p(2) & p(3) & \dots \\ 0 & 0 & Q(2) & p(3) & \dots \\ 0 & 0 & 0 & Q(3) & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}.$$

Обозначим $p_k(d_2|d_1)$ вероятность перехода с глубины d_1 на глубину d_2 за k шагов. Матрица переходных вероятностей за k шагов равна L^k [6]. Обозначим $p_k(d)$ суммарную вероятность перехода на глубину d за k шагов:

$$p_k(d) = \sum_{i=0}^{d} p(i)\, p_k(d|i).$$

Пусть $M << r$ — объем выборки, при котором встреча после k шагов происходит с большой вероятностью, например $1/e$. Вероятность встречи зависит от глубины. Встреча не произойдет, если ни на одной глубине в выборке не окажется двух одинаковых вершин. Если на глубину d попадут M_d вершин, то вероятность того, что среди них не будет двух одинаковых, равна $\exp\left(-\frac{M_d^2}{2r_d}\right)$. Если на глубины d_i попадут M_{d_i} вершин, то встреча произойдет с вероятностью

$$P = 1 - \prod_{d_i} \exp\left(-\frac{M_{d_i}^2}{2r_{d_i}}\right) \approx \sum_{d_i} \frac{M_{d_i}^2}{2r_{d_i}}. \qquad (3)$$

Поскольку $M_{d_i} = Mp_k(d_i)$, $r_{d_i} = rp(d_i)$, вероятность (3) можно записать в виде

$$P \approx \frac{M^2}{2r} \sum_{d_i} \frac{p_k^2(d_i)}{p(d_i)}. \qquad (4)$$

Таким образом, задача нахождения оптимального числа шагов k сводится к минимизации суммы $Mk + M\cdot\log_2 M$, при условии, что выражение (4) дает 1. Отметим, что минимизация Mk невозможна, так как она требует поиска максимума монотонно убывающей суммы $\frac{1}{k^2}\sum_{d_i}\frac{p_k^2(d_i)}{p(d_i)}$. Оптимум

для числа шагов примерно равен $k \approx \log_2 M \approx 0{,}5 \log_2 r$. Отсюда оптимальное значение

$$M \approx \sqrt{\frac{2r}{\sum\limits_{d_i} \frac{p_k^2(d_i)}{p(d_i)}}}. \qquad (5)$$

Вероятность $p_k(d)$ будем искать как сумму вероятностей путей длины k на глубину d, число таких путей равно $\binom{d}{k}$. Вероятность каждого пути равна произведению вероятностей соответствующих шагов.

Оценим оптимальное значение M снизу, положив $p(d_i) = \frac{2}{d_i^2}$ и положив $Q(d_i) = 1$. Путь длины k имеет не более k пропусков очередной глубины. Для вероятности $\pi^j(d)$ пути с j пропусками на глубину d имеет место неравенство $\pi^j(d) < p(d-j)\sigma_j(p(0), ..., p(d-1))$, где σ_j — j-я симметрическая функция. Положим

$$p_k^*(d) = p(d-k)\sum_{j=0}^{k}\sigma_j(p(0)\ldots p(d)). \qquad (6)$$

Тогда $p_k(d) < p_k^*(d)$.

Значения функции σ_j стремятся к нулю с ростом j. Кроме того, значение функции σ_j сходится с ростом d. Например, для точных значений вероятностей получается $\sigma_1(64) = 0{,}603$, $\sigma_1(256) = 0{,}624$, $\sigma_1(1024) = 0{,}630$, $\sigma_2(64) = 0{,}16$, $\sigma_2(256) = 0{,}172$, $\sigma_2(1024) = 0{,}176$, $\sigma_3(64) = 0{,}025$, $\sigma_3(256) = 0{,}028$, $\sigma_3(1024) = 0{,}029$. Сумму симметрических функций в (6) можно оценить сверху числом 2.

Подставим (6) в сумму в правой части (4) с учетом того, что $p_k(d_i) = 0$ при $i < k$ и $\sum\limits_{d=k}^{O(\sqrt{r})} p(d) = P(k)$. Получим:

$$\sum_{d_i=k}^{O(\sqrt{r})}\frac{p_k^2(d_i)}{p(d_i)} < \sum_{d_i=k}^{O(\sqrt{r})}\frac{4p^2(d_i-k)}{p(d_i)} \approx \frac{4}{P(k)}.$$

Отсюда при $k = \frac{\log r}{2}$ получаем $M > \sqrt{\frac{r}{\log r}}$, т.е. сложность алгоритма встречи на случайном дереве равна $O(\sqrt{r\log r})$ и всегда превышает сложность алгоритма Полларда.

Аналогичными рассуждениями можно показать, что при запоминании не конечных вершин после k шагов, а всех k вершин, встречающихся при спуске, алгоритм Полларда улучшить также невозможно.

4. Результаты эксперимента

Для проверки правильности полученных теоретически выводов был проведен эксперимент. В качестве случайного отображения использовалась функция $x_{i+1} = f(x_i) + x_i$, где x_i — блок данных длиной 48 бит, $f(x_i)$ — операция, аналогичная зашифрованию блока в режиме простой замены на 16 циклах по ГОСТ 28147–89, "+" — операция покоординатного сложения двоичных векторов.

Для случайного начального значения x_0 делалось k последовательных отображений указанного вида. Полученная база данных сортировалась и в ней искались равные элементы. Число шагов k изменялось в диапазоне от 1 до 1000. Сложность алгоритма вычислялась по формуле $S = Mk + M \log_2 M$, где M — объем базы данных, при котором происходила встреча.

Экспериментально получено, что оптимальное число шагов лежит в диапазоне 8÷32. С ростом числа шагов свыше 256 сложность алгоритма резко возрастает, практически рост сложности пропорционален числу шагов. Сложность алгоритма встречи на случайном дереве всегда превышает сложность алгоритма Полларда.

Кроме того, был проведен эксперимент, при котором запоминались не конечные вершины после k шагов, а все k промежуточных вершин и среди них искались равные. Сложность вычислялась по формуле

$$S = Mk + Mk \log_2(Mk).$$

Экспериментально подтверждено, что эта модификация алгоритма менее эффективна, чем алгоритм с запоминанием конечных вершин после k шагов.

Литература

1. Pollard J. Monte Carlo methods for index computatiom (mod p) // Math. Comp., v. 32, 1978, pp. 918–924.
2. Menezes A., van Oorchot P., Vanstone S. A handbook of applied cryptograhy. — CRC Press, 1996.
3. Ростовцев А. Г., Маховенко Е. Б. Введение в криптографию с открытым ключом. — Мир и Семья, Интерлайн, С.-Петербург, 2000.
4. van Oorchot P., Wiener M. Parallel collision search with cryptanalitic applications // Journal of Cryptology, v. 12, 1999, pp. 1–28.
5. Колчин В. Ф. Случайные отображения. — М.: Наука, 1984.
6. Феллер В. Введение в теорию вероятностей и ее приложения. — М.: Мир, 1984.