Гаусса. Оптимизация заключается в том, чтобы не менять столбцы, которые менялись на нескольких предыдущих итерациях; тем самым базисы, фиксируемые на итерациях, становятся более независимыми, что немного повышает вероятность найти вектор ошибки e.

2) На очередной итерации алгоритма Бернштейна — Ланг — Петерса ищется вектор ошибки e в виде линейной комбинации ровно $2p$ векторов базиса. Некоторым образом выбираются линейные комбинации из $2p$ векторов базиса, которые образуют векторы-кандидаты. Если вес какого-то вектора-кандидата равен t, то это и есть искомый вектор e. Для проверки необходимо вычислить веса всех векторов-кандидатов. Бернштейн, Ланг и Петерс предлагают считать вес каждого вектора, пока он не превысит t (дальше считать бессмысленно). Оптимизация заключается в том, чтобы отбросить все векторы-кандидаты, у которых среди первых a координат больше чем b координат принимают значение 1. Для остальных осуществить проверку так же, как её делают Бернштейн, Ланг и Петерс. Здесь a и b — новые параметры алгоритма. Смысл оптимизации заключается в следующем: на раннем этапе будет отброшено очень много неподходящих векторов-кандидатов, в то время как подходящий вектор-кандидат может быть отброшен с очень маленькой вероятностью.

Для представленного алгоритма ожидаемое количество битовых операций, необходимых для дешифрования сообщения, закодированного с помощью кодов Гоппы $(1024, 524, 50)$, равно $2^{60,1}$. Это на 27,5 % меньше, чем для алгоритма Бернштейна — Ланг — Петерса, самого быстрого из существующих алгоритмов неструктурной атаки на криптосистему Мак-Элиса. Тем самым осуществлено ещё большее приближение к теоретической оценке количества битовых операций при дешифровании сообщения.

ЛИТЕРАТУРА

1. *McEliece R. J.* A public-key cryptosystem based on algebraic coding theory // DSN Progress Report. January and February 1978. No. 42–44. P. 114–116.
2. *Finiasz M. and Sendrier N.* Security bounds for the design of code-based cryptosystems // Asiacrypt'2009. LNCS. 2009. V. 5912. P. 88–105.
3. *Bernstein D. J., Lange T., and Peters C.* Attacking and defending the McEliece cryptosystem // Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008. Cincinnaty, OH, USA. October 17–19, 2008. P. 31–46.

УДК 519.17, 004.056.2, 004.056.53

О ВЕРОЯТНОСТНЫХ ХАРАКТЕРИСТИКАХ СЛУЧАЙНЫХ ГРАФОВ, ПОРОЖДАЕМЫХ АЛГОРИТМАМИ ПОИСКА КОЛЛИЗИЙ КРИПТОГРАФИЧЕСКИХ ХЭШ-ФУНКЦИЙ

Г. А. Карпунин

Описывается теоретико-графовая модель некоторых алгоритмов поиска коллизий хэш-функций SHA-1 и RIPEMD, и в данной модели выводится точная формула средней трудоёмкости этих алгоритмов.

Ключевые слова: *криптографические хэш-функции, коллизии, случайные графы.*

В алгоритмах поиска коллизий некоторых хэш-функций семейства MDx (см., например, SHA-1 [1] и RIPEMD [2, 3]), встречается процедура $\mathcal{A}$, которую можно смоделировать случайным процессом Γ_N. Данный случайный процесс строит корневое

дерево G максимальной глубины N. При этом некоторые из вершин дерева G оказываются помеченными как плодоносящие, и только плодоносящие вершины могут иметь потомков. Процесс Γ_N считается успешным, если он построит дерево G глубины ровно N и последняя вершина на глубине N, которую сформирует процесс, окажется помеченной как плодоносящая.

У процесса Γ_N имеется два набора параметров $\{p_k\}_{k=N}^0$ и $\{n_k\}_{k=N}^1$, где $0 \leqslant p_k \leqslant 1$ и $n_k \in \mathbb{N}$. Формально процесс $\Gamma_N(\{p_k\}_{k=N}^0; \{n_k\}_{k=N}^1)$ описывается с помощью следующей рекурсивной процедуры. На вход процессу подается корневая вершина R. Сначала процесс с вероятностью p_N помечает её как плодоносящую. Если R оказалась непомеченной, то процесс завершает неуспешно свою работу и в качестве построенного графа возвращает лишь непомеченный корень R. Если же R оказалась помеченной как плодоносящая, то процесс строит её первого потомка R_1 и передает его в качестве корневой вершины производному процессу $\Gamma_{N-1}(\{p_k\}_{k=N-1}^0; \{n_k\}_{k=N-1}^1)$. Если производный процесс завершился успешно, то и процесс $\Gamma_N(\{p_k\}_{k=N}^0; \{n_k\}_{k=N}^1)$ также завершается успешно, иначе он строит второго потомка R_2 и для него снова запускает производный процесс $\Gamma_{N-1}(\{p_k\}_{k=N-1}^0; \{n_k\}_{k=N-1}^1)$. Всего может быть построено максимум n_N потомков $R_1, \ldots, R_{n_N}$. Если для каждого из них производный процесс завершился неуспешно, то и процесс $\Gamma_N(\{p_k\}_{k=N}^0; \{n_k\}_{k=N}^1)$ также завершается неуспешно. Пограничный процесс $\Gamma_0(p_0; \varnothing)$ никаких потомков не строит, он просто помечает с вероятностью p_0 корневую вершину как плодоносящую.

Обозначим через P_i вероятность успешного завершения процесса $\Gamma_i(\{p_k\}_{k=i}^0; \{n_k\}_{k=i}^1)$. Несложно показать, что для вероятностей P_i выполняется следующее рекуррентное соотношение:

$$P_i = p_i(1 - (1 - P_{i-1})^{n_i}), \tag{1}$$

при этом из определения очевидно, что $P_0 = p_0$. С помощью формулы (1) можно вычислить вероятность успеха P_N процедуры $\mathcal{A}$.

На практике [1–3] алгоритмы поиска коллизий запускают процедуру $\mathcal{A}$ несколько раз до первого успеха и в качестве меры эффективности всего алгоритма берётся его средняя трудоёмкость, которая равна величине $\mathbb{E}T(\mathcal{A})/P_N$, где $T(\mathcal{A})$ — трудоёмкость процедуры $\mathcal{A}$. В качестве меры трудоёмкости процедуры $\mathcal{A}$, в свою очередь, служит мощность множества вершин $V(G)$ построенного графа G. Таким образом, практически важной величиной является отношение $\mathbb{E}V(G)/P_N$, для вычисления и оценки которого доказана теорема 1.

Теорема 1. Имеют место следующие соотношения:

$$\frac{\mathbb{E}V(G)}{P_N} = \sum_{i=0}^{N} \frac{1}{P_i} \geqslant \lim_{\substack{n_k \to \infty, \\ k=1,\ldots,N}} \sum_{i=0}^{N} \frac{1}{P_i} = \sum_{i=0}^{N} \frac{1}{p_i}.$$

Из теоремы 1 следует, что для минимизации средней трудоёмкости алгоритмов поиска коллизий, использующих процедуру $\mathcal{A}$, параметры n_k необходимо выбирать как можно больше, если на них нет других ограничений.

ЛИТЕРАТУРА

1. *De Cannière C. and Rechberger C.* Finding SHA-1 characteristics: general results and applications // ASIACRYPT-2006. LNCS. 2006. V. 4284. P. 1–20.

2. *Wang X., Lai X., Feng D., et al.* Cryptanalysis of the hash functions MD4 and RIPEMD // EUROCRYPT-2005. LNCS. 2005. V. 3494. P. 1–18.
3. *Ермолаева Е. З., Карпунин Г. А.* Оценки сложности поиска коллизий для хэш-функции RIPEMD // Прикладная дискретная математика. Приложение. 2012. № 5. С. 43–44.

УДК 519.713

ОБ ОБРАТИМОСТИ КОНЕЧНЫХ АВТОМАТОВ С КОНЕЧНОЙ ЗАДЕРЖКОЙ

Д. А. Катеринский

Построены экспериментальные оценки доли обратимых, слабо обратимых и сильно обратимых конечных автоматов с конечной задержкой, из которых следует, что эта доля мала (до 3 %) для автоматов с близкими мощностями их алфавитов состояний и выходных символов и велика (более 80 %) для автоматов, у которых выходной алфавит в 4 раза мощнее входного и в 2 раза — внутреннего.

Ключевые слова: *конечные автоматы, слабая обратимость, обратимость, анализ обратимости, синтез обратных автоматов, доля обратимых автоматов.*

Рассмотрены автоматы, обратимые с нулевой задержкой, и автоматы, слабо или сильно обратимые с конечной задержкой. В первых функция выходов инъективна в каждом состоянии, во вторых входная последовательность восстанавливается с задержкой по выходной последовательности и начальному состоянию, а в третьих — только по выходной последовательности. Для каждого типа обратимости известны тест обратимости и алгоритм построения обратного автомата [1, 2].

В работе сообщается о программной реализации этих тестов и алгоритмов и об экспериментальных оценках доли обратимых автоматов всех типов. Полученные оценки приведены на рис. 1, где на оси абсцисс отмечена доля обратимых автоматов, на оси ординат — значения параметров автоматов, для которых проводилось исследование: m, n и k — мощности соответственно входного, внутреннего и выходного алфавитов автомата. В каждой точке оценки построены усреднением результатов вычислений для 10^4 примеров случайных автоматов. Результаты для доли обратимых автоматов с нулевой задержкой совпадают с теоретическими, вычисленными по формуле

$$d = \frac{(C_k^m \cdot m!)^n}{k^{mn}}.$$

Из рис. 1 видно, что:

1) доля обратимых автоматов мала (менее 3 %), если мощности входного, внутреннего и выходного алфавитов близки друг к другу или мощности внутреннего и выходного алфавитов меньше мощности входного алфавита;

2) доля обратимых автоматов высока (более 80 %), если мощность выходного алфавита много больше (более чем в 4 раза) мощности входного алфавита и больше (хотя бы в 2 раза) мощности внутреннего алфавита.