Криптоанализ с применением rainbow-таблиц

А.А. Семенов, 14.12.2011 (ИМ им. С.Л. Соболева СО РАН).

Поиск коллизий в хэш-функциях

Для начала мы опишем общую идею, лежащую в основе rainbow-таблиц, которую проиллюстрируем на примере задачи поиска коллизий в криптографических хэш-функциях.

Напомним, что коллизией первого рода для неизвестного $x \in \{0,1\}^*$ при хеш-преобразовании

$$\chi : \{0,1\}^* \to \{0,1\}^C, \; C \text{ – некоторая константа,}$$

называется такой $x' \in \{0,1\}^*$, что $\chi(x') = \chi(x)$. Грубо говоря, задача поиска коллизии первого рода состоит в том, чтобы, зная значение $\chi(x)$, найти такой x' (не обязательно совпадающий с x), что $\chi(x') = \chi(x)$.

Поиск коллизий в хэш-функциях

Данная задача возникает, например, при подборе пароля почтового ящика. Современные системы электронной почты хранят не собственно пароли, а их хеш-образы. Предполагается, что злоумышленник может найти такой хеш-образ, но использовать его непосредственно для входа в ящик он не может, поскольку системе надо самой вычислить хеш по некоторому входу, а потом сравнить то что получилось с имеющимся. В данной ситуации злоумышленнику достаточно найти такой x', что $\chi(x') = \chi(x)$.

Далее мы предполагаем, что известно некоторое значение $\chi(x)$ и известно n (такое, что $x \in \{0,1\}^n$). Также предполагается, что возможен полный перебор множества $\{0,1\}^n$ и вычисление значений $\chi(x)$ от его элементов. Мы пока не останавливаемся подробно на том, как это можно сделать. Отметим лишь, что для этой цели могут быть использованы супервычислители (в т.ч. распределенные вычислительные среды).

Пространственно-временной компромисс М. Хеллмана

Впервые идею «пространственно-временного компромисса», лежащего в основе rainbow-таблиц, предложил М. Хеллман в работе

M. E. Hellman. A cryptanalytic time-memory trade off // IEEE Transactions on Information Theory, IT-26:401–406, 1980.

Основная идея Хеллмана состояла в том, что помимо алгоритма вычисления функции χ можно использовать алгоритм вычисления т.н. «редукционной функции» ρ, действие которой «противоположно» действию χ в том смысле, что ρ по известному хеш-значению $\chi(x)$ строит некоторый $\tilde{x} \in \{0,1\}^n$. Функция ρ должна вычисляться очень быстро. Особо подчеркнем, что ρ – это не обратная к χ функция. Поэтому совсем не обязательно, что $\chi(\tilde{x}) = \chi(x)$ (это было бы слишком большой удачей).

Пространственно-временной компромисс М. Хеллмана

Далее при помощи отображений χ и ρ для всех x из $\{0,1\}^n$ строятся т.н. «цепочки» – последовательности суперпозиций χ и ρ некоторой фиксированной длины:

$$\chi(x) \to \rho(\chi(x)) \to \chi(\rho(\chi(x))) = \chi(x^2) \to \rho(\chi(\rho(\chi(x)))) \to \ldots \to \chi(\ldots(x)) = \chi(x^t) \qquad (1)$$

Далее мы предположим, что сгенерированы m цепочек вида (1). При этом все эти цепочки имеют различные первые и последние элементы. Основной момент: *в итоговой таблице для каждой цепочки хранятся только первый и последний элементы*. В предположении, что повторение элементов в цепочках незначительно, имеем некоторое покрытие исходного пространства ключей ключами, находящимися в построенных цепочках (но, подчеркнем, в таблице хранятся только первый и последний элементы таких цепочек).

Пространственно-временной компромисс М. Хеллмана

Пусть имеется некоторое хеш-значение y. Будем применять к y попеременно преобразования ρ и χ. Сделаем не более $t-1$ таких итераций. Если при этом мы не получим второго элемента некоторой пары из нашей таблицы, то можно сделать вывод, что исходный $x:\chi(x)=y$, а также все его коллизии оказались непокрытыми нашей таблицей.

Предположим, что мы получили

$$\rho(y)\to\chi(\rho(y))\to\ldots\to\chi(\ldots(\rho(y)))=\chi(\tilde{x}^t),$$

причем $(\tilde{x},\chi(\tilde{x}^t))$ – пара, хранящаяся в нашей таблице. В силу сказанного это означает, что данная пара представляет цепочку

$$\tilde{x}\to\chi(\tilde{x})\to\ldots\to\underbrace{\rho(\chi(\ldots))}_{x'}\to\chi(x')=y\to\rho(y)\to\chi(\rho(y))\to\ldots\to\chi(\ldots(\rho(y)))=\chi(\tilde{x}^t)$$

Очевидно, что x' – искомая коллизия. Таким образом, в данном случае мы можем обратившись к таблице, взять из нее соответствующий $\tilde{x}$ и досчитать «начальную часть» до искомой коллизии, используя попеременную суперпозицию χ и ρ.

Пространственно-временной компромисс М. Хеллмана

На первый взгляд описанная идея представляется очень продуктивной. Однако при ближайшем рассмотрении становятся видны ее многочисленные отрицательные стороны. Заметим, что, во-первых, необходим компромисс между длиной цепочки t и числом цепочек m (если один из них чрезмерно велик по отношению к другому, эффективность поиска падает). Далее мы полагаем, что некоторые такие параметры зафиксированы. В таблице каждая цепочка должна начинаться со своего уникального элемента – это уже накладывает ограничения на «произвольность построения». Предположим, тем не менее, что такая таблица построена.

Пространственно-временной компромисс М. Хеллмана

Используя несложные вероятностные соображения, можно показать, что если при этом начальные элементы цепочек «случайно распределены», то весьма высока вероятность того, что элемент x, с которого начинается некоторая цепочка C_1, является элементом другой цепочки – C_2, но находится при этом не в начале C_2. Однако в любом случае это означает, что часть элементов цепочек C_1 и C_2 совпадает. Более того, вся «конечная часть» цепочки C_2 после элемента x совпадает с начальной частью цепочки C_1.

Пространственно-временной компромисс М. Хеллмана

Такие ситуации (и им подобные) называют склеиванием (merging) цепочек. Очевидно, что если такие ситуации возникают часто, то большой процент ключей из $\{0,1\}^n$ вообще не покрывается нашей таблицей (то есть они вообще не присутствуют ни в одной цепочке). Вероятность покрытия таблицей произвольного ключа при определенных допущениях на характер редукционной функции (она должна иметь «достаточно случайные свойства») оценивается (это было сделано еще М. Хеллманом в 1980 г.) как

$$P_1 \approx \frac{1}{2^n} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(1 - \frac{i \cdot t}{2^n}\right)^{j+1}$$

здесь m – число цепочек в таблице, а t – длина цепочки.

Rainbow-таблицы

В 2003 году Philippe Oechslin в работе

P. Oechslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off," Proc. 23rd Ann. Int'l Cryptology Conf. (CRYPTO '03), pp. 617-630, 2003.

предложил модификацию идеи Хеллмана, известную теперь как rainbow-таблицы. В частности, он предложил использовать не одну редукционную функцию, а несколько и, соответственно, строить несколько таблиц по описанному выше принципу. Легко убедиться, что в таком случае вероятность покрытия произвольного ключа хотя бы одной из l таблиц с различными редукционными функциями (обладающими «хорошими» свойствами) можно оценить как

$$P_r \approx 1 - \left(1 - \frac{1}{2^n} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(1 - \frac{i \cdot t}{2^n}\right)^{j+1}\right)^l.$$

Rainbow-таблицы

Также несложно заметить, что цепочки в различных таблицах не могут склеиваться так, как это происходит в одной таблице, поскольку редукционные функции различны. Однако остается возможность «столкновений» (collide) цепочек, то есть ситуаций, когда одни и те же элементы все же встречаются в разных цепочках разных таблиц (правда, вразброс). Таким образом, rainbow-таблицы также не покрывают полностью всего ключевого пространства (при приемлемых параметрах на размеры таблиц), но имеют гораздо больший потенциал использования за счет подбора различных редукционных функций.

Далее мы рассмотрим возможности использования rainbow-таблиц в криптоанализе поточных шифров и конкретно в криптоанализе A5/1.

Применение rainbow-таблиц в криптоанализе

Как обычно мы считаем, что нам известен некоторый фрагмент ключевого потока и необходимо найти инициализирующую последовательность его породившую.

Итак, рассматриваем генератор

$$A : \{0,1\}^n \to \{0,1\}^M, M \geq n .$$

Так же, как и при поиске коллизий, используем одну или несколько редукционных функций вида

$$\rho : \{0,1\}^M \to \{0,1\}^n .$$

Строим цепочки вида

$$x^1 \to A(x) \to \rho(A(x)) = x^2 \to \ldots \to A(x^t),$$

пытаясь покрыть ими все пространство $\{0,1\}^n$. Затем организуем первый и последний элементы цепочек в виде таблицы (или нескольких таблиц при различных редукционных функциях).

Применение rainbow-таблиц в криптоанализе

Пусть $y \in \{0,1\}^M$ – некоторый ключевой поток, порожденный генератором A. Применим к нему попеременно суперпозицию функций ρ и A (не более $t-1$ итерации). Если при этом мы не получим ни для одной пары из нашей таблицы второго элемента, то делаем вывод, что таблица не покрывает искомый секретный ключ.

Предположим, что для некоторой пары $\left(\tilde{x}, A\left(\tilde{x}^t\right)\right)$ мы получили

$$\rho(y) \to A(\rho(y)) \to \ldots \to A(\ldots(\rho(y))) = A\left(\tilde{x}^t\right)$$

Тогда мы берем из таблицы $\tilde{x}$ и восстанавливаем соответствующую цепочку:

$$\tilde{x} \to A(\tilde{x}) \to \ldots \to \underbrace{\rho(A(\ldots))}_{x'} \to A(x') = y \to \rho(y) \to A(\rho(y)) \to \ldots \to A(\ldots(\rho(y))) = A\left(\tilde{x}^t\right)$$

Очевидно, что x' – искомый ключ, порождающий ключевой поток y.

Применение rainbow-таблиц в криптоанализе

За 8 лет, прошедшие с работы Oechslin –a, rainbow-таблицы претерпели ряд изменений. Как правило, понятие «rainbow-таблица» не воспринимается как набор таблиц, в котором каждая таблица соответствует отдельной редукционной функции, а рассматривается как одна таблица, в которой при построении цепочек редукционные функции чередуются в определенном порядке. При этом важным моментом является сокращение числа вычислений за счет различных «ухищрений», использующих специальные структуры данных. Мы приведем пример такого рода оптимизации, получившей название техники «особых точек» (*Distinguished Points, DP).*

Применение rainbow-таблиц в криптоанализе

Особые точки (DP) – это специального рода метки, которые позволяют существенно сократить вычисления при поиске x в таблице. Построение таблицы с DP осуществляется следующим образом. Мы генерируем очередную цепочку до тех пор, пока она либо не закончится особой точкой (например, 20 подряд идущими нулями), либо пока длина цепочки не достигнет $t_{\max}+1$. В таблице сохраняются только те цепочки, которые заканчиваются DP. При задании DP используются дополнительные соображения. Допустим, что наше преобразование (хэш-функция или генератор) и редукционные функции действуют из $\{0,1\}^{64}$ в $\{0,1\}^{64}$. Определим DP как 20 подряд идущих нулей. Рассмотрим произвольную цепочку, начинающуюся с некоторого $x^* \in \{0,1\}^{64}$ и заканчивающуюся некоторым $\tilde{x} \in \{0,1\}^{64}$, у которого последние 20 бит нули. Но в этом случае первые 44 бита данного $\tilde{x}$ и x^* идентифицируют конкретную цепочку (начинающуюся с x^* и заканчивающуюся $\tilde{x}$).

Применение rainbow-таблиц в криптоанализе

Допустим, что наша таблица построена и рассмотрим y, прообраз которого мы ищем. Мы можем применять к y редукционные функции и исходное преобразование и не заглядывать в нашу таблицу до тех пор, либо пока не получим особую точку, либо пока не превысим $t_{\max}$ (в этом случае искомое x не покрыто таблицей). Если мы получили особую точку, то смотрим на последнее $\tilde{x}$, заканчивающееся этой точкой, и находим соответствующее ему x^*, тем самым идентифицируя цепочку. После этого находим прообраз y в данной цепочке. Этот простой механизм позволяет существенно сократить вычислительные затраты на поиск в таблице. Помимо DP часто используются различные схемы варьирования редукционных функций, а также комбинации этих методов.

Применение rainbow-таблиц в криптоанализе

Построение rainbow-таблицы является вычислительно трудоемкой задачей и не под силу одному («обычному») человеку. В 1980 году, когда Хеллман высказал свои общие идеи по данному типу криптоаналитических атак, их практическое осуществление в отношении систем шифрования с 64-битным ключом было невозможно. Однако на данный момент с развитием вычислительных архитектур такие атаки стали вполне реальны.

Многочисленные примеры подбора паролей в системах, использующих различные алгоритмы хеширования, описаны в различных источниках. Если длина пароля не превышает 64 битов (8 символов в 8-битной ASCII-кодировке), то такие атаки на сегодняшний день вполне реальны. Практически всегда для этой цели используются rainbow-таблицы. Отметим, что данная атака фактически использует не уязвимость алгоритма, а малую длину пароля, являясь, таким образом, «продвинутой» разновидностью брутфорса.

Rainbow-таблицы в криптоанализе A5/1

В 2008 году авторы архитектуры COPACOBANA

T. Guneysu, T. Kasper, M. Novotny, C. Paar and A. Rupp, "Cryptanalysis with COPACOBANA," *IEEE Transactions on computers*, vol. 57, no. 11, 2008, pp. 1498–1513.

проанализировали ее вычислительные возможности для реализации различных вариантов «rainbow-метода», в том числе и применимых к криптоанализу поточных шифров. По их расчетам работа вычислителя COPACOBANA в течении 337 дней должна была позволить построить rainbow-таблицу объемом 7,49 терабайт, вероятность покрытия произвольного ключа в шифре A5/1 для которой составляет порядка 86%.

Rainbow-таблицы в криптоанализе A5/1

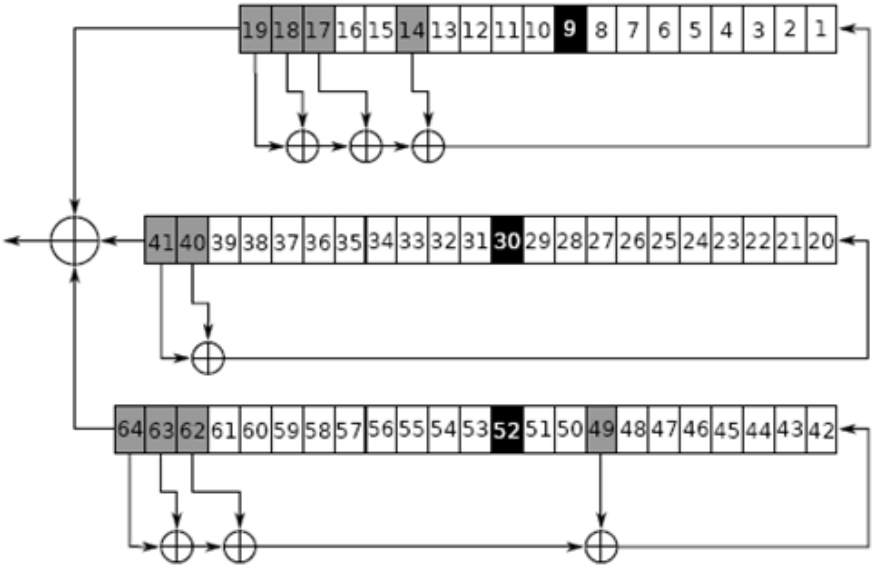
В декабре 2009 года Karsten Nohl и Chris Paget объявили об успешном построении rainbow-таблиц для криптоанализа A5/1 в рамках т.н. «A5/1 Cracking Project». Для построения таблиц они использовали ресурсы примерно 40 распределенных GPU-вычислителей в течение 3-х месяцев. Их таблицы занимают приблизительно 1,7 терабайт и доступны для скачивания посредством протокола BitTorrent (http://opensource.srlabs.de/attachments/download/41/a51_table_torrents.tgz).

По утверждению авторов проекта данные таблицы позволяют находить с вероятностью около 90% секретный ключ, используя знание 8 т.н. «бёрстов» (burst) ключевого потока, которые практически всегда можно «достать» из одного телефонного разговора.

Техника, которая при этом используется, задействует не только саму rainbow-таблицу, но и ряд весьма интересных деталей, на которых имеет смысл остановиться подробнее.

Rainbow-таблицы в криптоанализе

Широко известный генератор ключевого потока A5/1 используется для шифрования трафика в протоколе GSM. Собственно генератор выглядит следующим образом:



Примитивы генератора – 3 асинхронно сдвигаемых регистра сдвига с линейной обратной связью (РСЛОС, LFSR). Сдвиг каждого регистра в момент времени τ происходит, если значение серединного бита данного регистра совпадает со значением функции majority от значений серединных битов всех трех регистров.

Rainbow-таблицы в криптоанализе A5/1

Далее мы остановимся на особенностях протокола (а не генератора) шифрования A5/1. Данный протокол шифрует информацию т.н. «фреймами». Один фрейм состоит из 4 «бёрстов». Один бёрст – это 114 бит информации. Для шифрования каждого бёрста используется собственный «псевдоключ», который строится из секретного ключа (который мы ищем) и номера бёрста. Номер бёрста – это всегда 22 битное число. Из особенностей протокола вытекает следующий факт – перехватывая зашифрованный поток одного звонка, можно выделить в нем как минимум 8 бёрстов криптотекста, для которого известен исходный текст! Это происходит из-за несовершенства той части протокола, в которой телефоны пользователей обмениваются служебной информацией (оказывается, что в этих 8 бёрстах всегда передается заранее известная последовательность).

Rainbow-таблицы в криптоанализе A5/1

Таким образом, мы имеем следующую задачу. Нам известно 8 бёрстов ключевого потока. Каждый бёрст – это 114 бит выхода генератора A5/1, порожденного по псевдоключу данного конкретного бёрста. Причем собственно схема порождения бёрста состоит из трех этапов. Сначала происходит инициализация генератора, в результате чего в его регистрах оказывается псевдоключ. Затем генератор работает «вхолостую» 100 тактов (соответствующие порожденные биты просто отбрасываются). И уже затем порождает 114-битный бёрст. Нам необходимо по 8 известным бёрстам определить секретный ключ. Как это можно сделать при помощи rainbow-таблицы? Первое, что приходит в голову – зная произвольный (например, первый) бёрст, попробовать найти по таблице соответствующее состояние регистров, которое его породило. Однако это даже не псевдоключ! (поскольку генератор работал вхолостую 100 тактов). Вопрос: что делать в данной ситуации?

Rainbow-таблицы в криптоанализе A5/1

Предположим, что таблица выдала нам состояние регистров (скажем, B_1, B_2, B_3), которое породило анализируемый бёрст. Ключевым является следующий простой факт, справедливый не только для A5/1, но и для многих других поточных шифров. Мы совершенно четко уверены, что каждый регистр до момента начала генерации нашего бёрста был сдвинут не более 100 раз. Мы можем восстановить все эти гипотетические состояния (по 100 штук для каждого регистра), а затем опробовать все возможные их сочетания, которых $100^3 = 10^6$. Не так и много для современных вычислителей! Кроме того, на самом деле реально возможных таких сочетаний, конечно же, меньше. В результате такого анализа может найтись несколько «кандидатов в ключи», порождающих рассматриваемый бёрст. Тогда остальные бёрсты можно использовать для проверки и «отсева» ложных кандидатов.

Rainbow-таблицы в криптоанализе A5/1

Именно эта идея используется авторами A5/1 Cracking Project. Однако если поступать именно так, то есть пытаться восстановить ключ только на основе собственно бёрстов, вероятность успеха будет крайне мала! С целью ее повышения в A5/1 Cracking Project используется еще один (впрочем, задолго до них известный) прием. Он опять-таки применим к большей части поточных шифров и базируется на том факте, что каждый бит ключевого потока не зависит от предыдущих (а определяется только состоянием регистров на момент генерации). Таким образом, мы можем пытаться определить не сразу искомый секретный ключ, а состояние, которое привело к генерации некоторых M бит в бёрсте.

Rainbow-таблицы в криптоанализе A5/1

Из ранних работ Й. Голича известно, что для восстановления ключа в A5/1 достаточно 64 бит ключевого потока. Исходя из этого мы рассматриваем N бит ключевого потока как $N-M+1$ сдвигов (на один бит) M-битных фрагментов. При $N=114$ и $M=64$ имеем 51 такой сдвиг. Для каждого (из 51) фрагмента, состоящего из 64 битов, будем применять rainbow-таблицу и находить порождающее данный фрагмент состояние. Если таковое нашлось (большинство, разумеется, будет пропущено), то используем описанную выше процедуру «отматывания назад», результатом которой является некоторое множество кандидатов в ключи. Очевидно, что нам достаточно найти хотя бы один «удачный» фрагмент, то есть такой, что среди множества кандидатов в ключи, построенному (в результате «отматывания») по данному фрагменту, есть искомый ключ. Описанный подход известен как TMDTO (time-memory-data-tradeoff)-метод.

Rainbow-таблицы в криптоанализе A5/1

- Итак, использование всех перечисленных методов позволяет на таблицах A5/1 CP достигать 88,8% вероятности вычисления ключа в A5/1 по 8 бёрстам. Эта информация была экспериментально подтверждена.
- Полезные ссылки.

http://reflextor.com/trac/a51
http://www.ks.uni-freiburg.de/download/misc/practical_exercise_a51.pdf
http://srlabs.de/blog/wp-content/uploads/2010/07/100729.Breaking.GSM_.Privacy.BlackHat.pdf
http://events.ccc.de/congress/2009/Fahrplan/attachments/1519_26C3.Karsten.Nohl.GSM.pdf
http://www.ks.uni-freiburg.de/download/masterarbeit/SS11/09-betz-gsm/masterarbeit_johann_betz.pdf

СПАСИБО ЗА ВНИМАНИЕ!