Применение SAT Solvers для криптоанализа хеш-функций

Авторы: Илья Миронов и Lintao Zhang
Microsoft Research, Silicon Valley Campus
{mironov,lintaoz}@microsoft.com
Перевод: Ипполитов В.А.

Реферат

Несколько стандартных криптографических хэш-функций были взломаны в 2005 году. Некоторые существенные составные блоки этих атак хорошо поддаются автоматизации, посредством кодирования их в виде КНФ формул, которые находятся в пределах досягаемости современных SAT solvers. В дальнейшем будет показана эффективность этого подхода. В частности, возможно произвести полные коллизии для MD4 и MD5 когда даны только дифференциальные пути и примененяя (минимально измененные) имеющиеся на данный момент SAT solvers. Для лучшего понимания, это первый пример автоматизированного SAT solver для криптоанализа нетривиальных криптографических примитивов. Ожидается, что SAT solver будет найдено новое применение, такое как проверочный и тестирующий инструмент для практикующих криптоаналитиков.

1. Введение

Определители булевой выполнимости (SAT Solvers) добились значительного прогресса в последние десятилетия [MSS99, MM3 + 01, ES03]. Рекордная производительность современных SAT Solvers открывает новые перспективы для их применения там, где это считалось возможным лишь условно. Тем не менее, большинство из успешных реальных приложений SAT Solvers относятся к традиционным областям формальной проверки и ИИ. Далее будет рассмотрено применение SAT Solver дял криптоанализа хэш-функций.

Несколько приложений SAT Solvers для криптоанализа были описаны в литературе [Mas99, MM00, FMM03, JJ05]. Их стратегию можно рассматривать как поход «в лоб», в том смысле, что они не используют любые новые или существующие криптографические методы в своих атаках. Неудивительно, что эти усилия не дали ничего, представляющего интерес для криптологов.

Несмотря на предыдущее (возможно неудачные) попытки, существует убеждение, что SAT Solvers могут быть использованы в практическом криптоанализе.

Недавно несколько важных криптографических хэш-функций были показаны, как уязвимые к атакам, основанным на поиске коллизий [WY05, WYY05b]. Настоящие атаки состояли из нескольких шагов, каждый из которых включает в себя много ручной работы с битами. Достаточно сказать, что атака на простейшую функцию семейства, MD4, требует отслеживания более, чем 122 логических условий.

Далее будет показано, что SAT Solvers могут быть использованы для автоматизации некоторых элементов этих атак. В частности, что SAT Solvers могут устранить необходимость составления таблиц достаточных условий и

проектирование умных техник модификаций сообщений. Успешные атаки на MD4 и MD5 предполагают, что SAT Solvers могут быть ценным дополнением к инструментарию криптоаналитиков.

2 Теория и конструкция хэш-функций

Криптографические хэш-функции имеют важное значение для безопасности многих протоколов. Раннее применение хэш-функций в системах безопасности включают таблицы паролей [JKW74] и схемы подписи [RSA78, Lam79] и с тех пор практически любой криптографический протокол использует прямо или косвенно защитные хеш-функции в качестве основного блока.

Свойства, необходимые для безопасности хэш-функции отличаются и часто зависят от протокола в запросе. Тем не менее, свойство быть устойчивой к коллизиям признано «золотым стандартом» безопасности хэш-функции. Первое официальное определение устойчивости к коллизиям хэш-функции (CRHF) было дано Damgard'ом [Dam88]. Функция H называется устойчивой к коллизиям, если невозможно найти два разных входных x, y такие, что H (x) = H (Y). Так как любая хеш-функция имеет коллизии, гарантия устойчивости к коллизиям может быть только вычислительной.

Первый стандарт хеш-функции, MD4, был разработан Роном Ривестом [Riv91], его модифицированная версия MD5 вскоре после этого [Riv92]. Первая NIST-утвержденная хэш-функция, SHA (Secure Hash Algorithm), адаптированная к общей структуре (и даже некоторым константам!) MD4 [NIS93]; она была выведена из использования в 1995 году и заменена на новую версию под названием SHA-1 [NIS95], которая отличаются одним дополнительным обучением. Чтобы избежать путаницы, оригинальные SHA обычно называют SHA-0. В 2004 году две хэш-функции распространенно (и почти исключительно) использовались: MD5 и SHA-1. Справедливости ради надо сказать, что все эти функции принадлежат к одной семье, которая имеет схожие принципи проектирования.

Функция сжатия. Основным блоком конструкции CRHF является устойчивая к коллизиям функция сжатия, которая отображает вход фиксированной длины в более короткий выход фиксированной длины.

Ядро конструкции представляет собой блочный шифр, который определяется как функция двух входов E: $\{0, 1\}^k \times \{0, 1\}^n \mapsto \{0, 1\}^n$. Хотя E $(\cdot,\cdot)$ сжимает свой вход путем сопоставления $k + n$ бит в битах k, так как это тривиально обратимо. Тем не менее, следующее действие, называемое конструкцией Дэвис-Мейера, приводит CRHF F к предположению, что E является идеальным блочным шифром (например, E (x, •) и что это проиндексированы набор случайных перестановок на $\{0, 1\}^n$):

$$F (M, X) = E (X, M) \oplus M.$$

Среди нескольких методов построения блочных шифров, *сеть Фейстеля* на сегодняшний день является наиболее известным, являясь методом выбора для DES. Несимметричная сеть Фейстела является основой всех блочных шифров MDx и Shax семей. Это итерационный метод, который состоит из двух отдельных компонент: алгоритма расширения ключа и набора раундов функций. Сеть параметризуется количеством раундов r и размерами состояния. Для конкретности, пусть состояние состоит из четырех 32-разрядных слов (как в случае MD4 и MD5). Состояние идет через r раундов преобразований, пусть начальные, промежуточные и конечные состояния будут (a_i, b_i, c_i, d_i) для $i \in \{0,\ldots, r\}$. Алгоритм расширения ключа K отражает M через r раундов как $K(M) = w_0,\ldots,w_{r-1}$:

$$K : \{0, 1\}^n \rightarrow \underbrace{\{0, 1\}^{32} \times \cdots \times \{0, 1\}^{32}}_{r \text{ раз}}.$$

Раунд функций f_i: $\{0, 1\}^{128} \rightarrow \{0, 1\}^{32}$ для $i \in \{0, \ldots, r-1\}$ используются для обновления состояния. Один из примеров MD5 преобразования:

$$(a_{i+1}, b_{i+1}, c_{i+1}, d_{i+1}) \leftarrow$$
$$(d_i, b_i + (a_i + f(b_i, c_i, d_i) + w_i + k_i) \lll s_i, b_i, c_i).$$

Следует обратить внимание, что преобразование является обратимым, если ключ раунда w_i известен.

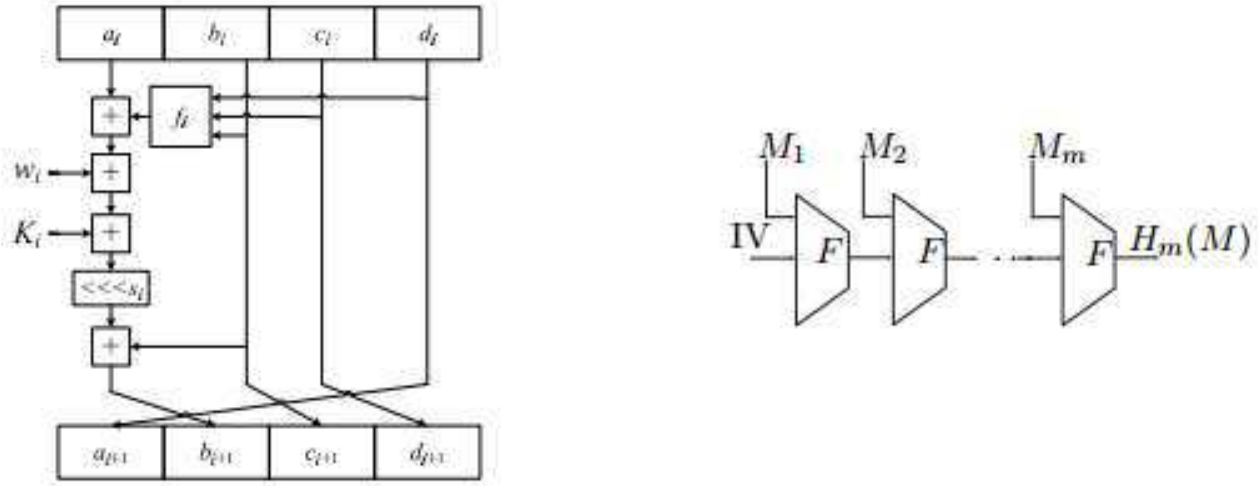

Рисунок 1 - Один раунд сети Фейстеля для MD5 и Меркле-Дамгард конструкции.

Парадигма Меркле-Дамгард. Ожидается, что CRHF принимает вход произвольной длины. Структура функций сжатия фиксированной длинны, рассмотренных выше, сохраняет свойство устойчивости к коллизиям. Этот метод называется конструкцией Меркле-Дамгарда [Mer90, Dam90] (См. Рисунок 1).

Общие атаки. Общие атаки на хеш-функции не обращают внимания на особенности их конструкции, они относятся к хеш-функциям как к черным ящикам и в целом обеспечивают верхнюю границу безопасности различных свойств шифрования.

Противопоставим две общих атаки на хэш-функции. Обе атаки с входными сообщениями $x \neq y$, такими, что $H(x) = H(y)$. В первой атаке два сообщения неограниченные, ее общая сложность составляет $2^n / 2$, где выходная длинна H - n бит. Цель второй атаки - найти сталкивающихся x и y, что $x = y \oplus \delta$ для некоторого фиксированного δ. Общая сложность этой атаки составляет 2^n.

Практические хэш-функции. В соответствии с номенклатурой указанной выше, структуры MD4, MD5, SHA-0 и SHA-1 хэш-функций следуют парадигме Меркле-Дамгарда, используя функцию сжатия, построенную через конструкцию Дейвиса-Мейера из блочных шифров несбалансированного типа сети Фейстеля.

Внутреннее состояние функции сжатия состоит из четырех 32-разрядных слов (a_i, b_i, c_i, d_i) для MD4 и MD5 и пяти 32-разрядных слов $(a_i, b_i, c_i, d_i, e_i)$ для SHA-0 и SHA-1. Поскольку раз мер внутреннего состояния является также размером выходной хэш-функции, выходная длинна MD4 и MD5 составляет 128 бит; SHA-0 и SHA-1 создают 160-битный выход. В MD4 применяется 48 раундов преобразований Фейстеля, MD5 имеет 64, а SHA-0, 1 используют по 80 раундов. Более подробную информацию о конструкции, в том числе о функциях раундов, алгоритмы расширения ключей, и константы, опущенные в интересах краткости, можно прочесть в [MvOV96] или соответствующих стандартах.

Обычно убеждение в безопасности MD5 и SHA-1 поддерживалось относительным отсутствием нападений на эти и связанные с ними функции. Хотя коллизии в MD4 были обнаружены в 1996 [Dob96a], некоторые недостатки были выявлены в MD5 и SHA-0 [dBB94, Dob96b, BC04] и было известно теоретическое нападение на SHA-0 [CJ98], не было найдено коллизий в MD5 и SHA-1 несмотря на более чем десять лет пристального внимания.

2005 год привел к радикальному изменению в понимании хэш-функций. Новая и улучшенная атака на MD4 [WLF+05], коллизии для MD5 и SHA-0 [WY05, WYY05b], и теоретические нападения на SHA-1 [WYY05a] были объявлены группой китайских ученых под руководством Xiaoyun Ван в двух последовательных конференциях. Независимо от их нападения на SHA-0 и уменьшено-раундовый SHA-1 были открыты Бихамом и соавт. [BCJ+05].

Большинство из этих атак концептуально просты, но их реализация, как правило, чрезвычайно трудоемкая. Несмотря на значительный интерес к обобщению атак и их применению в других контекстах, необходимое количество ручной работы может быть непреодолимым. Можно наблюдать, что некоторые компоненты атаки могут быть выражены как КНФ формулы и достаточно эффективно решены передовыми SAT Solvers.

3 Атаки на хэш-функции

3.1 Обозначения

В этой главе и далее, там где будут попытки найти коллизии между $M = (m_0, \ldots, m_{15})$ и $M' = (m'_0, \ldots, m'_{15})$, переменные w_i, a_i, b_i, c_i, d_i, e_i ссылаются на вычисление функции сжатия на входе M и их основные копии w'_i, a'_i, b'_i, c'_i, d'_i, e'_i к вычислению той же функции M'.

Будет рассмотрено два типа дифференциалов: в отношении XOR и в отношении разницы по модулю 2^{32}. Определим

$$\Delta^+ a_i = a_i - a'_i (mod\ 2^{32})) \text{ и аналогично } \Delta^+ w_i, \Delta^+ m_i, \Delta^+ b_i, \ldots [, \Delta^+ e_i]$$

И

$$\Delta^{\oplus} a_i = a_i \oplus a'_i \text{ и аналогично } \Delta^{\oplus} w_i, \Delta^{\oplus} m_i, \Delta^{\oplus} b_i, \ldots, [\Delta^{\oplus} e_i].$$

Пусть

$$\Delta^{\oplus}_i = (\Delta^{\oplus} a_i, \Delta^{\oplus} b_i, \Delta^{\oplus} c_i, \Delta^{\oplus} d_i, [\Delta^{\oplus} e_i]), \text{ и аналогично} \Delta^+_i.$$

Последовательность $\Delta^{\oplus}_0, \Delta^{\oplus}_1, \Delta^{\oplus}_2, \ldots$ (соотв., $\Delta^+_0, \ldots$) называется дифференциальным путем по отношению к XOR дифференциалу (соотв., на разницу по модулю 2^{32}). Далее, $\circ$ обозначает как + так и $\oplus$.

Собственно говоря, нападения Ван и соавт. зафиксировали точные значения для большинства из различных битов, относительно обоих дифференциалов. Приведенные кодировки в полной мере используют эту информацию.

3.2 Обзор атак

Концептуально, атаки на MDx и SHA-0, 1, имеют много общего. Самое удивительное, атаки решили, казалось бы, более сложную задачу, т. е. нахождение 512-битного сообщения, такого что $H(IV, M) = H(IV, M \circ \delta)$, где H является функция сжатия и δ зафиксировано. Как отмечалось ранее, общая сложность этой атаки (той, которая использует функцию как черный ящик) 2^n, где $n = 128$ или 160. Удачный выбор δ и набора умных методов для нахождения M, которые используют слабости функции сжатия, довели сложность атаки до менее чем 2^{42} оценки хэш-функции.

Концептуально атаки состоят из четырех этапов.

Этап I. Выбор $\Delta \circ m_0, \ldots, \Delta \circ m_{15}$.

Этап II. Выбор дифференциального пути $\Delta \circ_0, \ldots, \Delta \circ_{r-1}$, где r - число раундов ($r = 48$, 64 или 80).

Этап III. Поиск множества *достаточных условий* в сообщении $M = (m_0, \ldots, m_{15})$ и промежуточных переменных $a_i, \ldots, d_i$, которые гарантируют (с

высокой вероятностью), что сообщение пары M, $M' = (m_0 \circ \Delta^{\circ} m_0, \ldots, m_{15} \circ \Delta^{\circ} m_{15})$ Следует дифференциальному пути $\Delta^{\circ}{}_0, \ldots, \Delta^{\circ}{}_{r-1}$.

Этап IV. Выбор такого сообщения M, в котором все достаточные условия содержатся.

Атака может показаться нелогичной: вместо того, чтобы найти любые два сообщения, которые сталкиваются при хэш-функции, во-первых серьезно ограничивается пространство возможных пар сообщений, фиксируя их разницу, и, во-вторых, выбирая дифференциальный путь, ограничивается пространство возможных решений даже дальше.

Есть две причины, которые делают этот подход рабочим. Во-первых, дифференциальный путь тщательно подобран, чтобы максимизировать вероятность столкновения. Во-вторых, намеренно ограничив пространство решений, становятся известны некоторые важные свойства решения, которые позволят нам получить одно за итерационный процесс.

На первом этапе атаки, как правило, всё делается вручную или с применением некоторых эвристик, таких которые пытаются найти разницу с низким весом Хэмминга.

Второй этап является наиболее творческим этапом из всех четырех. Он в достаточной мере гибок, по той причине, что, так как функция раунда нелинейная, данная разница на входе может дать в результате много возможные различий в выходных данных. По иронии судьбы, атака обращает самое основое в безопасности хэш-функции – нелинейность раундовой трансформации – в свою пользу. Существует несколько ограничений, наложенных на дифференциальный путь. Во-первых, он должен быть возможным. Кроме того, должен быть подходящим, и, наконец, он должен способствовать третьему этапу атаки.

Третий этап тесно связан с предыдущим. Большинство достаточных условий естественным образом вытекают из свойств функций раундов.

Четвертый этап вычислительно наиболее интенсивный. Эффективная атака Вана и соавт. не была бы возможна с отказом на этой стадии. Действительно, очень маловероятно, что случайная пара сообщений M и M' последует по дифференциальному пути. В этом месте недавние нападения наиболее радикально отличаются от более ранних попыток. Идея состоит в том, чтобы начать с произвольным M сообщение, а затем аккуратно "сообщить" ему дифференциальный путь. Решающим вкладом Ван и соавт. было придумать набор инструментов исправления ошибок в дифференциальной пути одной за другой.

Существует утверждение, что SAT Solvers могут быть очень полезными для автоматизации третьего и четвертого етапов атак. Поскольку фактическая атака требует много итераций между вторым этапом и следующими двумя, любой метод, который позволит ускорить тестирование и проверку дифференциального пути становится полезным криптоаналитическим инструментом.