Криптоанализ ассиметричных криптосистем

Лекция

Ревизия: 0.1

Последняя модификация: Vlad Kovtun

Дата последней модификации: 04.01.2011 14:53:00

История изменений

28.11.2009 – Версия 0.1. Первичный документ. Владислав Ковтун

Содержание

Лекция 6. Криптоанализ ассиметричных криптосистем. Часть 1

Вопросы

- Введение.
- Криптоанализ хеш-функций
- Решение задачи факторизации.
- Решение задачи дискретного логарифмирования в поле чисел.
- Решение задачи дискретного логарифма в группе точек эллиптической кривой.
- Решение задачи дискретного логарифма для Якобиана гиперэллиптической кривой.

Введение

Практически все используемые алгоритмы асимметричной криптографии основаны задачах факторизации и дискретного логарифмирования в различных алгебраических структурах. Несмотря на то, что принадлежность этих задач к классу NP-полных задач не доказана, на сегодняшний день не найден полиномиальный алгоритм их решения. Они заключаются в решении математической задачи, положенной в основу асимметричного шифра.

Список таких задач и соответствующих им криптосистем приведен в лекции 5.

Криптоанализ хеш-функций

Для криптоанализа хеш-функций применяют метод коллизий. Кратко остановимся на его сути.

Электронно-цифровая подпись (ЭЦП) - это некоторая информация, позволяющая убедиться в подлинности документа, определить его автора, проверить время и дату подписания документа и т.д. (функциональный набор может быть различным).

Известно, что ЭЦП базируется на криптографическом образе документа (сообщения), который формируется криптрографической хеш-функцией.

Приведем классификацию атак на схемы электронной подписи:

- **Атака на основе известного открытого ключа** – самая слабая из атак, практически всегда доступна противнику;
- **Атака на основе известных подписанных сообщений** – в распоряжении противника имеется некоторое число пар (M, C), где M -некоторое сообщение, а C -допустимая подпись для него, при этом противник не может влиять на выбор M ;
- **Простая атака с выбором подписанных сообщений** – противник имеет возможность выбрать некоторое количество подписанных сообщений, при этом открытый ключ он получает после этого выбора;
- **Направленная атака с выбором сообщений** – выбирая подписанные сообщения, противник знает открытый ключ;
- **Адаптивная атака с выбором сообщений** – противник знает открытый ключ, выбор каждого следующего подписанного сообщения он может делать на основе знания допустимой подписи предыдущего выбранного сообщения.

Для контроля целостности сообщений, передаваемых по каналу связи, и генерации ЭЦП используются хэш-функции - математические или иные функции, которые принимают на входе строку переменной длины (называемую прообразом) и преобразуют ее в выводную сроку фиксированной (обычно меньшей) длины, называемую значением хэш-функции или сверткой. Более формально определение:

$A = \{A_1, \ldots, A_m\}$ -алфавит, A^*-множество слов конечной длины в алфавите A, A^l-множество слов длины l, l-длина значения хэш-функции. Пусть определена функция $H : A^* \to A^l$, которая обладает тем свойством, что значения функции H на словах, которые даже при отличии друг от друга только в одном знаке дают значительно отличающиеся образ. Тогда, получив по каналу связи сообщение и его образ, можно вычислить образ от сообщения, сравнить с полученным значением и проверить, не было ли сообщение искажено при передаче. Если функция H зависит также от симметричного ключа $k \in K$, то, помимо проверки целостности, добавление образа к сообщению подтверждает истинность сообщения. Такой способ подтверждения истинности называется кодом аутентификации (Message Autentification Code - MAC).

Однако такое подтверждение истинности еще не является электронной подписью. Подтверждение истины называется подписью, если ее могут проверить все, не знающие секретного ключа. Для того чтобы код аутентификации стал электронной подписью сообщения, необходимо использовать хэш-функции с дополнительными свойствами (например, на основе асимметричной криптосистемы). Если $k \in K$ -личный ключ, k' - открытый ключ, E_k и $D_{k'}$ - функции зашифровывания и расшифровывания соответственно, то электронной подписью документа $M \in A^*$ называется $C = D_{k'}(H(M))$.

Функции E_k и $D_{k'}$ - взаимно обратные, поэтому для проверки подписи под документом M достаточно вычислить $\widetilde{X} = H(M)$ и $\overline{X} = E_k(C)$. Если $\widetilde{X} = \overline{X}$, то автором документа может являться только обладатель личного ключа k. Перечислим основные свойства хэш-функций:

1. Если $M \neq M'$ хотя бы в одном знаке, то $H(M) \neq H(M')$ примерно на половине знаков. Могут найтись такие $M \neq M'$, что $H(M) = H(M')$, но выбор таких M и M' случайно маловероятен, а подбор труден.
2. Для произвольной строки $\alpha \in A^l$ трудно подобрать $M \in A^*$ такое, что $H(M) = \alpha$.
3. $H(M)$ вычисляется быстро.

Стойкость схемы ЭЦП зависит от стойкости используемых криптоалгоритмов и хеш-функций. Основная атака на хэш-функции это метод коллизий. Пусть B-мошенник, а D-лицо, подвергающееся атаке. B готовит два документа M и M' так, что документ M пользователь D охотно подпишет, а B заинтересован в подписи D под документом M'. Тогда, подписав у D документ M, B получает подпись *C*, которая представляет собой некоторую функцию F от образа сообщения: $C = F(H(M))$. Если M' выбрано так, что $H(M) = H(M')$, то B может предъявить пару (M', C): тогда атака удалась.

Каким образом можно выбрать два таких сообщения? По свойству 2, подбор осуществить сложно.

Оказывается, с большой вероятностью злоумышленник может реализовать подбор такого сообщения. Такой подбор основан на задаче о днях рождения (будет описан далее в разделе «Метод встречи посередине»). B выбирает два нужных сообщения M и M', при этом D может подписать сообщение M, но не подпишет сообщение M'. Варьируя в сообщениях интервалы, шрифты, формат или внося в текст незначительные изменения, B получает n пар вариантов M и M' *без* изменения их смысла. И, следовательно, n пар соответствующих им значений хэш-функций:

$$H(M_1) \text{ и } H(M_1')$$

$$H(M_2) \text{ и } H(M_2')$$

…

$$H(M_n) \text{ и } H(M_n')$$

Сообщения M_1, ..., M_n отличаются слабо, а их хэш-функции -значительно, т.е. можно считать, что значения хэш-функций выбираются случайно, равновероятно и независимо друг от друга. Обозначим $N = |A^l| = |A|^l$, $l \to \infty$; тогда при выборе $n \to t\sqrt{N}$ ($t > 0$-некоторая константа) вероятность того, что имеется пара сообщений

M_i и M_i', удовлетворяющих условию $H(M_i) = H\left(M_i'\right)$, вычисляется по формуле $p = \left(1 - \exp\left(-t^2\right)\right)\left(1 + o(1)\right)$. Поиск двух таких сообщений можно выполнить с использованием метода «встречи посередине» или метода Полларда.

Метод «встречи посередине»

Другой популярный метод криптоанализа - алгоритм «встречи посередине», требует такого же объема памяти, как метод Полларда, но при этом поддается эффективному распараллеливанию. Например, для логарифмирования в группе порядка p при параллельной работе n процессоров, где $n >> p$, время работы алгоритма уменьшается в n раз. Данный метод криптоанализа основан на «*парадоксе дней рождения*». Известно, что если считать, что дни рождения распределены равномерно, то в группе из 24 человек с вероятностью 0,5 у двух человек дни рождения совпадут. В общем виде, этот парадокс формулируется так: если $a\sqrt{b}$ предметов выбираются с возвращением из некоторой совокупности размером b, то вероятность того, что два из них совпадут, равна $1 - \exp\left(-\frac{a^2}{2}\right)$ (в описанном частном случае $b = 365$ - количество дней в году, $a\sqrt{b} = 24$, т.е. $a \approx 1{,}256$).

Если множество ключей криптоалгоритма замкнуто относительно композиции, т.е. для любых ключей k_i и k_j найдется ключ k_r такой, что результат шифрования любого текста последовательно на k_i и k_j равен криптограмме этого же числа на k_r, т.е. $F\left(k_j, F\left(k_i, x\right)\right) = F\left(k_r, x\right)$, то можно воспользоваться этим свойством. Пусть нам нужно найти ключ k_r. Тогда для нахождения ключа k_r необходимо найти эквивалентную ему пару ключей k_i, k_j.

Пусть известен открытый текст x и его криптограмма y. Для текста x строим базу данных, содержащую случайное множество ключей k' и соответствующих криптограмм $w = F\left(k', x\right)$, и упорядочиваем ее по криптограммам w. Объем базы данных выбираем $O\left(\sqrt{\left|\{k'\}\right|}\right)$, где $\left|\{k'\}\right|$-мощность множества ключей k'. Затем подбираем случайным образом ключи k'' для расшифровки текстов y и результат расшифрования $v = F\left(k'', y\right)$ сравниваем с базой данных. Если текст v окажется равным одной из криптограмм w, то ключ $k'k''$ эквивалентен искомому ключу k. Этот метод также применим, если множество ключей содержит достаточно большое подмножество, являющееся полугруппой.

Обозначим $\alpha = \left|\{k\}\right|$ - общее количество возможных ключей k. Временная сложность метода составляет $O\left(\log\alpha\sqrt{\alpha}\right)$. Множитель $\log\alpha$ учитывает сложность сортировки. Требуемая память равна $O\left(\log\alpha\sqrt{\alpha}\right)$ бит, или $O\left(\sqrt{\alpha}\right)$ блоков (предполагается, что длина блока и длина ключа различаются на ограниченную константу).

Алгоритм является вероятностным. Однако существуют и детерминированный аналог этого алгоритма «giant step - baby step» (больших и малых шагов) с такой же сложностью, предложенный американским математиком Д. Шенксом.

Решение задачи факторизации

Задача

Пусть дан целое число $N > 1$, которое не является простым, тогда существует целое число a, такое, что $1 < a < N$ и $a \mid N$. Задача состоит в том, что бы найти такое a или доказать, что оно равно 1. В криптографических приложениях используют число N, состоящее из двух простых множителей соизмеримой величины, другими словами они имеют соизмеримые двоичные длины.

Сложность криптоанализа

Задача факторизации большого целого числа N на множители принадлежит к классу субэкспоненциальных алгоритмов, сложность которых составляет $L_N(\alpha, \beta) = \exp\left((\beta + o(1))(\log N)^{\alpha} (\log\log N)^{1-\alpha}\right)$.

Сложность алгоритма GNFS (General Number Field Sieve) - решета поля чисел общего вида составит $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)$.

Сложность алгоритма SNFS (Special Number Field Sieve) - решета поля чисел специального вида $N = a^b + c$ составит $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$.

Сложность решения задачи с использованием квантового компьютера с количеством $O(\log N)$ кубитов составит $O(\log N)$.

Известно, что для факторизации чисел двоичной длины до 1024 бит, наиболее приемлемым считается алгоритм ECM (Elliptic Curve Method) с использованием эллиптических кривых, для чисел большего размера принято использовать GNFS алгоритм.

Описание

Поиском эффективных способов разложения целых чисел на множители, занимаются давно. Наиболее очевидный метод разложения числа n на множители – перебор простых делителей не превышающих $\sqrt{n}$.

Существует также другой переборный метод (метод Ферма), который основан на представлении числа n в виде разности квадратов:

$$n = a^2 - b^2 = (a + b)(a - b).$$

Ферма предложил, вычисляя $\gcd(n, a - b)$, попытаться найти нетривиальный делитель n. Он предложил способ, позволяющий найти требуемое представление. Если разлагаемое число имеет два не очень различающиеся по величине множителя, этот способ позволяет определить их быстрее, чем простой перебор делителей. Лежандр обратил внимание на то, что при таком подходе достаточно получить сравнение

$$a^2 \equiv b^2 \bmod n \qquad (1).$$

Не каждая пара чисел, удовлетворяющая ему, позволяет разложить n на множители. Для нахождения чисел, связанных соотношением (1), Лежандр использовал *непрерывные дроби*. Существуют некоторые усовершенствования метода Ферма.

По сложности, как простой переборный метод, так и метод Ферма оцениваются величиной $O\left(\sqrt{n}\right)$, однако последний метод может оказаться эффективнее, если делители числа n близки друг к другу.

Для достаточно больших чисел n время работы таких алгоритмов становится неприемлемым. Прежде чем приступать к факторизации таких чисел, следует убедиться в том, что они действительно составные. Для этого лучше всего использовать один из вероятностных тестов на простоту, например, алгоритм Миллера—Рабина.

Методы факторизации с экспоненциальной сложностью для больших чисел используются в сочетании с субэкспоненциальными методами факторизации, которые будут описаны далее, и применяются, как правило, для предварительного отделения небольших простых делителей у факторизуемого числа.

Существуют и более эффективные методы разложения чисел на множители – это методы, имеющие субэкспоненциальную оценку сложности. Так, вероятностный алгоритм Ленстры для факторизации целых чисел с помощью эллиптических кривых имеет среднюю оценку временной сложности $\exp\left(((2 + o(1))\log p \log\log p)^{\frac{1}{2}}\right)\log^2 n$, где p - минимальный простой делитель n. При замене p на $\sqrt{n}$ получаем субэкспоненциальную оценку сложности $L_n\left[\frac{1}{2}; 2\right]$. Этот метод может быть эффективно использован для отделения небольших простых делителей. Преимуществом также является использование небольшого объема памяти.

Рассмотрим еще один субэкспоненциальный алгоритм факторизации – метод Диксона. Пусть $n \in \mathbf{N}$ -число, которое мы хотим разложить на множители, $0 < a < 1$ -некоторая постоянная, значение которой будет определено ниже. *Факторной базой* будем называть множество простых чисел p_i таких, что $2 \le p_i \le L^a$ (где $L = \exp\left(\left((2+o(1))\log p \log\log p\right)^{\frac{1}{2}}\right)$). Обозначим символом k количество простых чисел в факторной базе, а $Q(m)$ -наименьший неотрицательный *вычет в классе* $m^2 \pmod n$. Случайным перебором ищем числа $m_1, \ldots, m_{k+1}$, такие, что $1 < m_i < n$, $Q(m_i) = p_1^{\alpha_{i,1}} \cdots p_k^{\alpha_{i,k}}$, $i = \overline{1, k+1}$. $m_i^2 \equiv Q(m_i) \pmod n$ являются гладкими числами, т.к. могут быть представлены в виде произведения небольших простых чисел.

Обозначим $\vec{v_i} = (\alpha_{i,1}, \ldots, \alpha_{i,k}) \in \mathbf{Z}^k$ - векторы показателей в разложении $Q(m)$ на множители ($\mathbf{Z}^k$ -векторное пространство столбцов длины $n \in \mathbf{N}$ с координатами из $\mathbf{Z}$).

Решая систему линейных уравнений $x_1\vec{v_1} + \cdots + x_1\overrightarrow{v_{k+1}} = \vec{0} \pmod 2$ в векторном пространстве $\mathbf{Z}_2^k$ (k-мерное булево пространство), находим значения $x_1, \ldots, x_{k+1} \in \{0, 1\}$, не все из которых равны нулю (такое решение существует, поскольку число уравнений k меньше числа неизвестных).

Для вычисленных значений $x_1, \ldots, x_{k+1}$ справедливо соотношение:

$$\left(m_1^{x_1} \ldots m_{k+1}^{x_{k+1}}\right)^2 \equiv p_1^{\sum_{i=1}^{k+1} x_i \alpha_{i,1}} \cdots p_k^{\sum_{i=1}^{k+1} x_i \alpha_{i,k}} \pmod n.$$

Обозначим $X = m_1^{x_1} \ldots m_{k+1}^{x_{k+1}}$, $Y = \prod_{j=1}^{k} p_j^{\left(\sum_{i=1}^{k+1} x_i \alpha_{i,j}\right)/2}$, числа $\left(\sum_{i=1}^{k+1} x_i \alpha_{i,j}\right)/2$ -целые по определению x_i. Получаем соотношение $X^2 \equiv Y^2 \pmod n$. Далее проверяем условие $1 < \gcd(X \pm Y, n) < n$. В случае успеха мы разложили n на множители (вероятность успеха не меньше $\frac{1}{2}$). В случае неудачи возвращаемся в начало алгоритма и ищем другие значения m_i.

При выборе границы факторной базы равной $L^2 = \exp\left(2(\log n \log\log n)^{\frac{1}{2}}\right)$ временная сложность алгоритма Диксона оценивается как $L_n\left[\frac{1}{2}; 2\right]$. Существует несколько стратегий, позволяющих повысить эффективность алгоритма Диксона:

- Стратегия LP (использование больших простых чисел).
- Стратегия PS (применение алгоритма Полларда—Штрассена).
- Стратегия EAS (стратегия раннего обрыва).

В алгоритме Бриллхарта-Моррисона случайный выбор m из алгоритма Диксона, заменяется на детерминированное определение очередного значения m, для которого мы ищем разложение $m^2 \pmod n$ на простые множители из факторной базы. Этот выбор m делается с помощью непрерывных дробей для числа $\sqrt{m}$. Данный метод факторизации был наиболее популярным до появления в 1981 г. алгоритма квадратичного решета Померанца. Покажем его основную идею.

В качестве факторной базы выберем ограниченные по величине некоторым параметром - простые числа p_i, такие, что $\left(\frac{n}{p_i}\right) = 1$, где $\left(\frac{\alpha}{\beta}\right)$ - так называемый *символ Якоби*.

Обозначим через s количество таких чисел. Обозначим $m = \lfloor \sqrt{n} \rfloor$, $Q(t) = (t+m)^2 - n \equiv H(t)^2$, где $H = t + \lfloor \sqrt{n} \rfloor$. При малых целых значениях t величина $Q(t)$ будет сравнительно невелика. На следующем шаге, вместо того чтобы перебирать числа t и раскладывать соответствующие значения $Q(t)$ на множители, алгоритм сразу же отсеивает «ненужные» значения t, оставляя только те, для которых $Q(t)$ имеет делители среди элементов базы разложения:

$A = Q(t) = \prod_{j=1}^{s} p_j^{\gamma_j}$,

т.е. $Q(t)$, раскладывается в базе разложения. Тогда, обозначая $B = H(t)$, получаем сравнение $B^2 = A^2 (\mathrm{mod}\, n)$, и, накопив достаточно много таких соотношений, проводим исключение переменных и строим соотношение $X^2 \equiv Y^2 (\mathrm{mod}\, n)$ так же, как в алгоритме Диксона.

Эвристическая оценка сложности усовершенствованного алгоритма квадратичного решета составляет $L_n\left[\frac{1}{2}; 1\right]$ арифметических операций. Рекордное значение для факторизованных этим методом чисел составляет 129-значное RSA-число n. Метод квадратичного решета следует применять для факторизации, если число n не превосходит 10^{110}. Для чисел большей величины следует использовать алгоритмы решета числового поля. Решето числового поля, по сути, не является алгоритмом: это метод вычисления, состоящий из нескольких этапов, и каждый из этих этапов обслуживается несколькими алгоритмами. Описание этого метода слишком объемно, чтобы уместиться в рамках данной лекции. На сегодняшний день алгоритмы решета числового поля и квадратичное решето Померанца – самые быстрые из известных алгоритмов факторизации больших чисел.

Решение задачи дискретного логарифма

В данном случае под задачей дискретного логарифма подразумевается целый класс задач, которые отличаются друг от друга лишь алгебраическими структурами, образующими группы. Далее дадим наиболее общее определение данной задачи. Пусть дана конечная циклическая группа $\langle G \rangle$ порядка n, образующий элемент P и произвольный элемент группы Q. Задача нахождения дискретного логарифма состоит в поиске такого целого числа x, что $Q = xP$. Отметим, что это число приведено по модулю n. Рассмотрим задачи дискретного логарифмирования (DLP) на конкретных алгебраических структурах, которые образуют группы.

Задача дискретного логарифмирования считается более сложной, чем задача факторизации. Если будет найден полиномиальный алгоритм ее решения, станет возможным и разложение на множители (обратное не доказано).

Дискретный логарифм в мультипликативной группе конечного поля

Описание задачи

Пусть даны групповые элементы g (генератор поля) и h, тогда под **решением DLP** будем понимать решение уравнения $h = g^l$ относительно l или доказательство того, что решения не существует. Необходимым условием является $h \in \langle g \rangle$, т.е. $l \in [0, \mathrm{ord}(g) - 1]$ и $h^{\mathrm{ord}(g)} = 1$, где $g, h \in \mathbf{GF}(q)$, n - порядок элемента образующего группу $n = \mathrm{ord}(g)$, l - секретный ключ, h - открытый ключ.

Сложность криптоанализа

В таблице 1 приведем оценки сложности решения DLP в поле.

Таблица 1. Оценки сложность решения DLP в поле

№	Название алгоритмов	Поле	Сложность	Тип
1	Обобщенное решето поля чисел	$\mathbf{GF}(p)$	Требуется $O\left(\exp\left((64/9)^{1/3} + o(1)\right)(\ln p)^{1/3}(\ln \ln p)^{2/3}\right)$ групповых операций.	Д
2	Обобщенное решето поля чисел	$\mathbf{GF}(2^m)$	Требуется $O\left(\exp\left(1.587 m^{1/3}(\ln m)^{2/3}\right)\right)$ групповых операций.	Д
3	Алгоритм Adleman-Coppersmith	$\mathbf{GF}(2^m)$	Требуется $O\left(\exp\left((c + o(1)) m^{1/3}(\log m)^{2/3}\right)\right)$ групповых операций, где $c \approx 1.4$.	Д

Д – детерминированный алгоритм, дающий точное решение.

Index-calculus

Последние достижения теории вычислительной сложности показали, что общая проблема логарифмирования в конечных полях не может считаться достаточно прочным фундаментом. Наиболее эффективные на сегодняшний день алгоритмы дискретного логарифмирования имеют уже не экспоненциальную, а субэкспоненциальную временную сложность. Это алгоритмы «index-calculus», использующие базу разложения. Первый такой алгоритм был предложен Адлеманом и имеет временную сложность $L_n\left[\frac{1}{2}; c\right]$ при вычислении дискретного логарифма в простом поле $\mathbf{GF}(p)$, где $L_n[\gamma; c] = \exp\left((c + o(1))(\log^{\gamma} N \log^{1-\gamma} \log n)\right)$, где $0 < \gamma < 1$, $c = const$, $c > 0$. Идея использования базы разложения применялась и ранее, например. На практике алгоритм Адлемана оказался недостаточно эффективным; Копперсмит, Одлыжко и Шреппель предложили алгоритм дискретного логарифмирования COS с эвристической оценкой сложности операций $L_n\left[\frac{1}{2}; 1\right]$. Алгоритм решета числового поля, предложенный Широкауэром, при $p > 10^{100}$ работает эффективнее различных модификаций метода COS; его временная сложность составляет порядка $L_p\left[\frac{1}{3}; \left(\frac{64}{9}\right)^{\frac{1}{3}}\right]$ арифметических операций.

Основная идея методов «index-calculus» заключается в том, что если:

$$\prod_{i=1}^{m} l_i = \prod_{j=1}^{n} y_j ,$$

для некоторых элементов конечного поля $\mathbf{GF}(p)$, то

$$\sum_{i=1}^{m} \log_g l_i \equiv \sum_{j=1}^{n} \log_g y_j \bmod(p-1). \qquad (2)$$

Получив достаточно много соотношений (2) (причем хотя бы одно из них должно включать элемент a, для которого $\log_g a$ известен), можно решить систему линейных уравнений относительно неизвестных $\log_g l_i$ и $\log_g y_j$ в кольце вычетов $\mathbf{Z}_{p-1}$ при условии, что количество неизвестных в уравнениях не слишком велико.

Самый простой подход к генерации соотношений вида (1) – выбрать произвольный элемент $a \in \mathbf{Z}_p$, вычислить $u = g^a \pmod p$ и с помощью перебора попытаться найти числа, удовлетворяющие соотношению:

$$u = \prod_{i=1}^{k} p_i^{\alpha_i} ,$$

где p_i -простые числа, такие, что $p_i < B$ для некоторой границы B. Если удалось найти такие числа, то u является гладким элементом с границей гладкости B.

Выделяются два основных этапа в работе алгоритмов:

- **на первой**, подготовительной стадии:
- формируется база разложения и на ее основе генерируется система линейных уравнений в кольце $\mathbf{Z}_{p-1}$;
- вид факторной базы (множество простых чисел, неприводимых многочленов или других объектов) и способы получения матрицы системы зависят от выбранного алгоритма.
- На **второй стадии** (которая является общей для рассматриваемых алгоритмов) требуется получить решение этой системы.

Интенсивные предварительные вычисления для каждого поля достаточно выполнить только один раз. Затем можно быстро вычислять различные дискретные логарифмы. Таким образом, все субэкспоненциальные методы дискретного логарифмирования в конечных полях сводятся к этой задаче решения систем линейных уравнений в кольцах вычетов.

Метод Полларда

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. Приведем

схематическое описание метода Полларда. Пусть на некотором конечном множестве M определено случайное отображение f. Применим это отображение поочередно ко всем элементам $x \in M$. Соединим пару $x, y \in M$ дугой, ведущей из x в y при условии, что $y = f(x)$. Полученный граф является **волновым графом первого рода**, или **функциональным**. Каждая компонента связности функционального графа представляет собой деревья, вырастающие из цикла. В графе случайного отображения почти все вершины лежат на одном дереве. При обращении стрелок случайное дерево описывается критическим ветвящимся процессом. Для случайного отображения f **длина** цикла и высота дерева в среднем равны $O\left(\sqrt{|M|}\right)$.

Для нахождения ключа в криптоалгоритме, основанном на задаче логарифма в некоторой группе, достаточно решить задачу о встрече на графе случайного отображения. Для этого из двух разных стартовых точек x_0', x_0'' строятся траектория до входа в цикл. Затем одна из двух конечных точек, лежащих в цикле, фиксируется, а траектория другой продолжается до встречи с фиксированной точкой. Для функции f и точки входа x_0 длина траектории составляет $O\left(\sqrt{|M|}\right)$ шагов. Типичный вид этой траектории содержит предельный цикл длины $O\left(\sqrt{|M|}\right)$ и отрезок до входа в цикл примерно такой же длины. В качестве индикатора замыкания траектории Полларда предложил использовать равенство $x_i = x_{2i}$, где x_i - i-я точка траектории для входа x_0. Всегда найдется значение i, при котором будет выполняться это равенство. Причем значение индекса i не превышает суммы длины пути до входа в цикл.

В качестве иллюстрации рассмотрим, каким образом можно использовать метод Полларда для нахождения коллизии (двух аргументов, дающих одинаковое значение хэш-функции). Такими аргументами будут элементы множества M, стрелки от которых под действием хэш-функции f ведут в точку входа в цикл. Параметром алгоритма является число v, определяющее доступную память (объем памяти ограничен величиной $O(v)$).

Практически алгоритм сводится к нахождению точки входа в цикл и может быть сформулирован следующим образом:

1. Войти в цикл, используя равенство $x_i = x_{2i} = t$.

2. Измерить длину цикла m, применяя последовательно отображение f к элементу t до получения равенства $f^m(t) = t$.

3. Разбить цикл m на v интервалов одинаковой или близкой длины и создать базу данных, запомнив и упорядочив начальные точки интервалов.

4. Для стартовой вершины п.1 выполнять одиночные шаги до встречи с какой-либо точкой из базы данных п.3. Отметить начальную и конечную точки интервала, на котором произошла встреча.

5. Стереть предыдущую и создать новую базу данных из v точек, разбив интервал, на котором произошла встреча, на равные по длине части, запомнив и отсортировав начальные точки интервалов.

6. Повторить процедуры пп.4,5 до тех пор, пока не получится длина интервала, равная 1. Вычислить точку встречи в цикле, вычислить коллизию как пару вершин, одна из которых лежит в цикле, а другая - нет.

Этот алгоритм требует многократного выполнения $O\left(\sqrt{|M|}\right)$ шагов до входа в цикл и выполнении сортировки базы данных. На каждом этапе при создании новой базы данных длина интервала сокращается в v раз, то есть количество повторов равно $O\left(\log_v |M|\right)$. Если $v << \sqrt{|M|}$, то сложностью сортировки базы данных можно пренебречь. Тогда сложность алгоритма равна $O\left(l\sqrt{|M|}og_v|M|\right)$.

Метод Полларда также применяется для решения задачи логарифма на циклической группе, поиска частично эквивалентных ключей и в некоторых других случаях.

Применительно к задаче логарифмирования этот метод позволяет отказаться от использования большого объема памяти по сравнению с методом встречи посередине. Кроме того, пропадает необходимость сортировки базы данных. В результате временная сложность по сравнению с методом встречи посередине снижается на множитель $O\left(\log_v |M|\right)$. Сложность этого метода в данном случае составляет $O\left(\sqrt{|M|}\right)$ шагов и требует памяти объема $O(1)$ блоков.

Дискретный логарифм в группе точек эллиптической кривой над конечным полем

Описание задачи

Пусть даны точки $P, Q \in E(\mathbf{GF}(q))$, тогда **решением ECDLP** будем понимать решение уравнение $Q = lP$ относительно l или доказательстве того, что решение не существует. Необходимым условием является $Q \in \langle P \rangle$, т.е. $l \in [0, \mathrm{ord}(P) - 1]$, причем $\mathrm{ord}(P) \cdot Q = O$, где P - базовая точка, образующая группу, - порядок базовой точки $n = \mathrm{ord}(P)$, l - секретный ключ, Q - открытый ключ, O - точка на бесконечности.

Сложность криптоанализа

Согласно приведенных в [4] результатов, сложность решения ECDLP для стандартизированных кривых, посредством метода ρ-Pollard, составляет $O\left(\left(\ln^c h\right)\sqrt{\pi\, h/2}\right)$ групповых операций, где h - наибольший простой делитель $\mathrm{ord}(P)$, c - небольшая константа.

Таблица 2. Соответствие двоичной длины ключей симметричных шифров, RSA и криптосистем построенных на ЭК над простыми и двоичными полями

Двоичная длина ключа симметричного алгоритма шифрования	Название симметричного алгоритма	Двоичная длина ключа RSA	Двоичная длина простого числа p для базового поля $\mathbf{GF}(p)$	Степень расширения m для базового поля $\mathbf{GF}(2^m)$
80	SKIPJACK	1248	192	163
112	Triple-DES	2432	244	233
128	AES Small	3248	256	283
192	AES Medium	7936	384	409
256	AES Large	15424	521	571

Сейчас, наиболее активные исследования производятся в области криптоанализа, которая позволит существенно снизить объемы вычислений для взлома конкретной криптосистемы (программно и/или аппаратно реализованной), посредством анализа информации полученной косвенным путем (Side Channel Attacks). Указанные подходы рассматривались ранее на лекции по криптоанализу симметричных криптосистем.

Дискретный логарифм в якобиане гиперэллиптической кривой над конечным полем

Описание задачи

Пусть дана гиперэллиптическая кривая $C(\mathbf{GF}(q))$ рода g и приведенные дивизоры $D_1, D_2 \in \mathbf{J}(\mathbf{GF}(q))$, под **решением HCDLP** будем понимать решение уравнения $D_2 = lD_1$ относительно l или доказательстве, что решение не существует, где $\mathbf{J}(\mathbf{GF}(q)) = D^0/\mathrm{P}$ - якобиан, P - множество основных дивизоров, D^0 - множество дивизоров степени 0, n - порядок дивизора D_1, т.е. $n = \mathrm{ord}(D_1)$, l - секретный ключ, D_2- открытый ключ [4].

Сложность криптоанализа

Наиболее удачным для решения HECDLP считается алгоритм Index-calculus. Некоторые идеи, при решении DLP, могут быть применимы и для вычисления HECDLP в якобиане $\mathbf{J}_C(k)$ гиперэллиптической кривой C, рода g над полем $k = \mathbf{F}_q$ в мнимой квадратичной форме. Если род g значительно больше по сравнению с q, тогда субэкспоненциальное

временем работы вероятностного алгоритма с известным $n = q^g$ обозначают $O(L_n[c])$, где $L_{q^g}[c] = \exp\left((c + o(1)\sqrt{g \log q \log g \log q}\right)$ для некоторой положительной константы c.

Наиболее эффективным алгоритмом решения задачи дискретного логарифма в Якобиане ГЭК принято считать алгоритм Theriault . Различные его варианты позволяют получить сложности $O\left(g^5 q^{2-2/(g+1)+\varepsilon}\right)$ и $O\left(g^5 q^{2-4/(2g+1)+\varepsilon}\right)$ [4].

Квантовые вычисления

На сегодняшний день, (анонсирован 17 февраля 2007 года) промышленный вариант адиабатический (псевдо) квантовый компьютер **Orion** предложенный компанией D-Wave Systems (www.dwavesys.com). Первый вариант содержал 16-кубитов.

Литература

1. Криптографическая защита информации в АСУ СН. Курс лекций. В.И. Долгов. ХВУ. 1998.

2. Криптографическая защита информации в информационных системах. Курс лекций. И.Д. Горбенко. ХНУРЭ. 2002.

3. Криптопреобразования с открытым ключом. Текущее состояние. Обзор. В.Ю. Ковтун. 2006. URL: http://www.nrjetix.com/r-and-d/publications/

4. Ковтун В.Ю., Збитнев С.И., Шевченко Д.В., Гиневский А.М. Исследование алгоритмов решения задачи дискретного логарифма на эллиптической и гиперэллиптической кривых // Восточно–Европейский журнал передовых технологий. –2004. –Вып. №6 (12). –С. 155–167. URL: http://www.nrjetix.com/r-and-d/publications/