Содержание

Взлом системы RSA

Ушаков Константин*

1 Описание системы RSA

Криптосистема RSA, изобретенная Роном Ривестом (Ron Rivest), Ади Шамиром (Adi Shamir) и Леном Адлеманом (Len Adleman), была впервые представлена в выпуске *Scientific American* в 1977 году. Эта криптосистема используется, в основном, для защиты личной информации и подтверждения подлинности электронных данных. В наши дни RSA реализована во многих коммерческих системах, например, для защиты сетевого трафика между веб-сервером и браузером или для проверки авторства электронной почты. Она необходима в системах, где требуется шифрования паролей, и потому является сердцем систем с оплатой посредством электронных кредитных карточек. Короче говоря, RSA часто используется в приложениях, в которых сохранность электронных данных имеет решающее значение.

Система RSA проверяется «на прочность» многими исследователями с момента первой публикации. Хотя 20 лет исследований привели к большому числу крайне интересных атак, ни одна из них не представляет большой угрозы. Они скорее иллюстрируют опасность неправильного использования RSA. Однако, на самом деле, надежное использование RSA — нетривиальная задача. Наша цель — провести краткий обзор некоторых из этих атак и описать математический аппарат, на котором они основываются. На протяжении всего обзора мы будем придерживаться стандартных обозначений и использовать имена Алиса и Боб для упоминания двух абстрактных общающихся между собой партнеров. Мы используем имя Марвин, чтобы обозначить злоумышленника, желающего подслушать общение Алисы и Боба или подделать их документы.

Мы начнем с описания упрощенного варианта шифрования с помощью RSA.

Пусть $N = pq$ — произведение двух больших простых чисел одного размера (по $n/2$ бит каждое). Обычно N занимает $n = 1024$ бит, то есть 309 десятичных знаков, а сомножители — 512 бит.

$$N = p * q;\ p, q \in \mathbb{P};\ p, q \simeq 2^{n/2}.$$

Пусть e и d — целые числа, удовлетворяющие соотношению

$$ed \equiv 1 \pmod{\phi(N)},$$

*Законспектировал лекцию **Зайцев Андрей**

где $\phi(N) = (p-1)(q-1)$ — порядок мультипликативной группы $\mathbb{Z}_N^*$. Мы назовём N *модулем RSA*, e — экспонентой кодирования, d — экспонентой раскодирования.

$$e, d \in \mathbb{Z} : \ ed \equiv 1 \pmod{\phi(N)}, \ \phi(N) = (p-1)(q-1) = |\mathbb{Z}_N^*|$$

Пара $< N, e >$ называется открытым (public) ключом. Как подсказывает само название, он общеизвестен и используется для шифрования сообщений.

Пара $< N, d >$ называется закрытым (или секретным, private) ключом, который известен только адресату зашифрованных сообщений. Секретный ключ позволяет расшифровывать зашифрованный текст.

Сообщение — целое число $M \in \mathbb{Z}_N^*$ [1]. Чтобы зашифровать M, вычисляем $C = M^{ed} \equiv M \pmod{N}$. Чтобы расшифровать зашифрованные данные, законный получатель сообщения вычисляет $C^d \pmod{N}$. Действительно, $C^d = M^{ed} \equiv M \pmod{N}$, откуда последнее неравенство вытекает из теоремы Эйлера.

Шифровка: $C = M^e \mod N$ - шифрованное сообщение.

Дешифровка: $C^d = M^{ed} = M \pmod{N}$

Определим функцию $RSA : x \mapsto x^e \pmod{N}$. Если d известно, то функцию можно легко обратить, используя вышеприведенное равенство. Назовем d *отмычкой*, позволяющей обращать нашу функцию. В нашем исследовании мы изучим сложность обращения функции RSA, когда отмычка d не известна; обозначим это как *взлом RSA*. Строго говоря, по данной тройке $< N, e, C >$ нужно узнать, насколько трудно вычислить корень степени e из C по модулю $N = pq$, когда разложение N на множители не известно. Так как $\mathbb{Z}_N^*$ — конечное множество, можно перебрать все элементы $\mathbb{Z}_N^*$, пока не найдётся нужное M. К сожалению, временная сложность такого алгоритма порядка N, то есть зависит экспоненциально от размера ввода (который порядка $\log_2 N$). Нас же интересуют алгоритмы со значительно меньшей временной сложностью, а именно, порядка n^c, где $n = \log_2 N$, а c — некоторая маленькая константа (скажем, меньшая пяти). Такие алгоритмы хорошо применимы на практике. В этой работе мы будем называть их *эффективными*.

В этом обзоре мы в основном изучаем функцию RSA, а не саму криптосистему как таковую. Грубо говоря, сложность обращения функции RSA подразумевает, что по данной тройке $< N, e, C >$ злоумышленник не может восстановить исходный текст M. Однако криптосистема должна быть устойчивой к самым разным изощренным атакам и, например, не должна давать *никакой* информации об исходном M (т.н. семантическая безопасность, semantic security). Мы не будем рассматривать такие атаки — просто укажем, что наша функция RSA семантически не защищена. По тройке $< N, e, C >$ можно, например, сосчитать символ Якоби M по N, зная C. Чтобы добиться семантической безопасности, в сообщение M добавляют случайные данные.

Функция $RSA : x \mapsto x^e$ — пример *односторонней функции с отмычкой* (trapdoor one-way function). Её можно легко вычислить, но (как мы знаем)

[1] понятно, что последовательность байт, в которых хранится текстовая или иная информация, может быть представлена сколь угодно большим целым числом

нельзя эффективно обратить без использования отмычки d (исключая особые случаи). Примером использования таких функций являются цифровые подписи. Чтобы подписать сообщение M, применим частный ключ $< N, d >$ к M и получим подпись $S = M^d \mod N$. Теперь каждый может проверить, что $< M, S >$ — именно наше сообщение: $S^e = M \mod N$.

Цифровые подписи: Получаем подпись для M: $S = M^d \pmod N$. Тогда по паре $< M, S >$ определяем: $S^e \pmod N = M$, т.е. S — это наша подпись.

Итак, чтобы взломать RSA, необходимо знать разложение N на множители. Несмотря на то, что алгоритмы для решения этой задачи постоянно совершенствуются, их вычислительная сложность слишком велика для того, чтобы они могли применяться на практике. Но сводятся ли все атаки к задаче разложения на сомножители? Покажем, что разглашение отмычки d и разложения M на множители эквивалентны.

Факт 1. Пусть $< N, e >$ — открытый ключ RSA. По закрытому ключу $< N, d >$ можно эффективно разложить на множители $N = pq$ и обратно, зная разложение N, можно эффективно найти d.

Доказательство.

$\Longrightarrow$:

Из разложения N на простые сомножители можно узнать $\phi(N)$. Зная e, можно вычислить d.

$\Longleftarrow$:

Обратно, с помощью d можно N разложить на множители. Зная d, вычисляем $k = de - 1$.

Из определения d и e следует, что k — кратное $\phi(N)$, которое является чётным:

$$\phi(N) = (p_1 - 1)(p_2 - 1)\dots(p_j - 1) > 2, \quad p_i \in \mathbb{P}.$$

Тогда

$$k = 2^t r, \quad r \not\vdots 2, \ t \geq 1.$$

$\forall g \in \mathbb{Z}_N^* \quad g^k = 1 \Rightarrow g^{k/2}$ — корень из 1 по модулю N. По китайской теореме об остатках, 1 имеет четыре квадратных корня по модулю $N = pq$: ± 1 и $\pm x$, так что $x \equiv 1 \pmod p$ и $x \equiv -1 \pmod q$. Зная хотя бы один из последних двух квадратных корней, можно найти разложение N на множители, посчитав НОД $(x - 1, N)$.

Выбранный случайно элемент $g \in \mathbb{Z}_N^*$ с вероятностью не менее $1/2$ (из всех возможностей выбора) принадлежит последовательности

$$g^{k/2}, g^{k/4}, \dots, g^{k/2^t} \pmod N$$

и является корнем из 1. Все элементы данной последовательности можно вычислить за время $O(n^3), n = \log_2 N$. $\square$

2 Элементарные атаки

Начнем с описания некоторых старых элементарных атак. Они иллюстрируют явное непонимание RSA. Существует много таких атак, мы же ограничимся двумя примерами.

Общий модуль [2]

Чтобы избежать генерирование различного модуля $N = pq$ для каждого пользователя, можно использовать одно N для всех. Допустим, некий администратор предоставляет i-тому пользователю уникальную пару e_i, d_i, из которой последний формирует открытый ключ $< N, e_i >$ и закрытый ключ $< N, d_i >$.

На первый взгляд, все работает правильно: зашифрованный текст $C = M^{e_a} \pmod N$, предназначенный для Алисы, не может быть расшифрован Бобом, так как у него нет d_a. Однако это неверно, и рассмотренная система небезопасна. Из факта 1 видно, что Боб может использовать свои экспоненты e_b, d_b, чтобы разложить на сомножители модуль N. После этого, он может восстановить закрытый ключ Алисы по открытому ключу e_a. Таким образом, модуль RSA *никогда* не должен использоваться более чем одним лицом.

Ослепление [3] Пусть $< N, d >$ — закрытый ключ Боба, а $< N, e >$ — его открытый ключ. Предположим, что Марвину нужно поставить подпись Боба на сообщение $M \in \mathbb{Z}_N^*$; но Боб не дурак и ни за что не подпишет M. Тогда Марвин может сделать следующее: выбрать случайное $r \in \mathbb{Z}_N^*$ и получить $M' = r^e M \mod N$, а затем попросить Боба подписать (якобы) случайное сообщение. Теперь Боб поставит свою подпись S' на совсем невинное сообщение M'. После этого Марвин без труда вычисляет $S = S'/r \mod N$ и получает подпись Боба S для исходного M:

$$S' = (M')^d \mod N$$

$$S^e = (S')^e / r^e = (M')^{ed} / r^e \equiv M'/r^e = M \pmod N.$$

Этот метод, который называется ослеплением (blinding), позволяет Марвину получить нужную подпись путем обмана Боба: Боб ставит подпись на случайное "ослепленное"сообщения, не зная, что за сообщение он на самом деле подписывает. Эта атака не представляет угрозы, так как в большинстве схем к сообщению добавляется случайный "мусор", который делает переход от M' к M невозможным. Хотя мы представили ослепление как атаку, на самом деле, это полезное свойство RSA для предоставления анонимности электронных платежей: "электронные деньги"позволяют покупать товар, не указывая конкретно покупателя.

3 Low Private Exponent

Маленькая экспонента раскодирования

[2] Common modulus
[3] Blinding

Чтобы уменьшить время дешифровки (или генерирования подписи), можно пользоваться не любым сколь угодно большим d, а некоторым маленьким. Так как возведение в степень по модулю (modular exponentiation) занимает время порядка $\log_2 d$, уменьшение значения d даёт выигрыш, по крайней мере, в степень числа 10 (для 1024-битного модулю). К сожалению, как показал М. Винер (M. Wiener), использование небольшого d приводит к полному поражению криптосистемы.

Теорема (М. Винер). Пусть $N = pq$, $q < p < 2q$, $d < \frac{1}{3}N^{1/4}$. По открытому ключу $< N, e >$, $ed \equiv 1 \pmod{\phi(N)}$, Марвин может эффективно восстановить d.

Доказательство. Доказательство основывается на приближении непрерывными дробями. Т.к. $ed \equiv 1 \pmod{\phi(N)}$, существует k, т. что $ed - k\phi(N) = 1$. Тогда:

$$\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}.$$

Отсюда, $\frac{k}{d}$ — приближение для $\frac{e}{\phi(N)}$. Хотя Марвин не знает значение $\phi(N)$, он может его приблизить с помощью N. Действительно, т.к.

$$\phi(N) = N - p - q + 1$$

и

$$p + q - 1 < 3\sqrt{N},$$

имеем

$$|N - \phi(N)| < 3\sqrt{N}.$$

Подставим N вместо $\phi(N)$:

$$\left|\tfrac{e}{N} - \tfrac{k}{d}\right| = \left|\tfrac{ed - k\phi(N) - kN + k\phi(N)}{Nd}\right| = \left|\tfrac{1 - k(N - \phi(N))}{Nd}\right| \leq \left|\tfrac{3k\sqrt{N}}{Nd}\right| = \tfrac{3k}{d\sqrt{N}}.$$

$$k\phi(N) = ed - 1 < ed$$
$$e < \phi(N)$$
$$k < d < \frac{1}{3}N^{1/4}.$$

Получаем неравенство:

$$\left|\frac{e}{N} - \frac{k}{d}\right| \leq \frac{1}{d\sqrt[4]{N}} < \frac{1}{2d^2}.$$

Это классическая оценка приближения. Число дробей $\frac{k}{d}, d < N$, которые приближают $\frac{e}{N}$, ограничено $\log_2 N$. Фактически мы их находим как подходящие дроби для $\frac{e}{N}$, увеличивая отрезки этих цепных дробей.

Таким образом, остаётся только вычислить $\log N$ подходящих дробей цепной дроби $\frac{e}{N}$. Среди них будет $\frac{k}{d}$. Так как $ed - \phi(N) = 1$, $НОД(k, d) = 1$ и потому $\frac{k}{d}$ — правильная дробь.

По этому алгоритму можно найти секретной ключ d за линейное время.

□

Т.к. N обычно длиной в 1024 бит, то чтобы защититься от этой атаки, требуется d длиной 512 бит, что не выгодно для маломощных устройств. Тем не менее, можно воспользоваться следующими методами, которые позволяют быстро проводить расшифровку и невосприимчивы к этой атаке.

Увеличение e. Будем использовать открытый ключ $< N, e' >$, где $e' = e + t\phi(N)$, t - некоторое большое число. Очевидно, e' может быть использовано вместо e для шифровки сообщения. Тогда k в доказательстве уже не маленькое, несмотря на малость d (допустим $e' > N^{1.5}$), и атака невозможна. К сожалению, увеличение e увеличивает время кодирования.

Использование китайской теоремы об остатках. Выберем d, т. что малы d_p и d_q (скажем, длиной 128 бит):

$$d_p = d \pmod{p-1}$$

$$d_q = d \pmod{q-1}.$$

Тогда быстрое декодирование сообщения C достигается следующим образом:

1) вычисляем $M_p = C^{d_p} \mod p$, $M_q = C^{d_q} \mod q$;

2) с помощью китайской теоремы об остатках вычисляем $M \in \mathbb{Z}_N$, т. что $M = M_p \mod p$ и $M = M_q \mod q$. $M = C^d \mod N$, как нам и нужно. Изюминка заключается в том, что хотя d_p и d_q малы, значение $d \mod \phi(N)$ велико, т.е. порядка $\phi(N)$. Поэтому атака Винера здесь неприменима.

4 Low Public Exponent

Маленькая экспонента кодирования

Чтобы уменьшить время шифрования или проверки подписи, желательно использовать маленькое e. Наименьшее возможное значение — 3, однако рекомендуется использовать $e = 2^{16} + 1 = 65537$ (для защиты от некоторых атак). При использовании значения $2^{16} + 1$ на проверку подписи требуется 17 умножений ($\forall e \leq \phi(N)$ возможно около 1000). В отличие от атак из предыдущего раздела, атаки, ориентированные на использование маленького e не всегда действенны.

Теорема Копперсмита [4]

Самые мощные атаки на маленькую экспоненту кодирования основываются на теореме Копперсмита, которая имеет много приложений, одно из которых атака Хастеда (Hastad).

Теорема 1. *Пусть $N \in \mathbb{Z}$ и $f \in \mathbb{Z}[x]$ — нормированный многочлен степени d. $X = N^{\frac{1}{d} - \varepsilon}$, $\varepsilon \geq 0$. Тогда по паре $< N, f >$ Марвин эффективно найдет все целые числа $|x_0| < X|$, т. что $f(x_0) = 0 \mod N$.*

Атака Хостеда. Представим, что Боб рассылает зашифрованное сообщение M адресатам $P_1, P_2, \ldots, P_k$. У каждого адресата свой ключ $< N_i, e_i >$. Допустим, что M меньше всех N_i-ых. Для шифровки M используется каждый

[4] Coppersmith's Theorem

открытый ключ, i-тое зашифрованное сообщение отправляется P_i-ому адресату. Для простоты считаем, что $e_i = 3$. Тогда Марвин может восстановить M, если $k \geq 3$.

Действительно, узнаем C_1, C_2, C_3, где

$$C_1 = M^3 \mod N_1$$

$$C_2 = M^3 \mod N_2$$

$$C_3 = M^3 \mod N_3.$$

Применим китайскую теорему об остатках и получим $C' = M^3 \mod (N_1 N_2 N_3)$. Так как $M^3 < N_1 N_2 N_3$, то сравнение переходит в обычное равенство, и, вычислив $\sqrt[3]{C'}$, узнаем M.

Защита от атаки Хастеда. Нужно генерировать шифрованные сообщения следующим образом:

$$f_i(M) = (i \cdot 2^i + M) \mod N.$$

Тогда

$$C_i = (f_i(M))^e \mod N.$$

И задача сводится к нахождению корней многочлена большой степени.