СРАВНИТЕЛЬНЫЙ АНАЛИЗ РАЗЛИЧНЫХ МОДИФИКАЦИЙ КРИПТОСИСТЕМ RSA

Цыбулько К. Д.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Иванюк А. А. – доктор технических наук,

профессор кафедры информатики

Криптосистема RSA с несколькими простыми числами и RSA с возведением в степень — это варианты RSA, которые позволяют улучшить производительность алгоритма дешифрации. В этой работе были представлены и проанализированы оба алгоритма. Так же были смоделированы системы шифрования и экспериментальные результаты ускорения сравнивались с математическими расчётами. В дополнение, были представлены оптимальные вариации, которые позволяют достичь достаточной производительности, при этом не ухудшая безопасность систем.

RSA [1] является наиболее широко используемой криптосистемой с открытым ключом. Используется для защиты веб-трафика, электронной почты и некоторых беспроводных устройств. Поскольку RSA основан на модульной арифметике больших чисел, он может быть медленным в определенных условиях. Например, при работе в сильно загруженной сети сервера, расшифровка RSA значительно снижает количество запросов SSL в секунду, с которыми сервер может справиться. Как правило, производительность RSA улучшается с помощью специального оборудования.

В этой статье будут рассмотрены два простых варианта RSA, предназначенных для ускорения процесса дешифрации. Важно подчеркнуть, что алгоритмы имеют обратную совместимость: система, использующая один из этих алгоритмов, должна иметь возможность взаимодействия с системами, созданными для стандартного RSA.

Первая модификация — это использование нескольких простых чисел при вычислении *N*. Алгоритм генерации ключа принимает в качестве входных данных параметр безопасности *N* и дополнительный параметр *b*. Он генерирует пару ключей RSA следующим образом:

1) Генерация *b* различных простых чисел $p_1, \dots, p_b$ каждая длиной $\left[\frac{n}{b}\right]$ бит. Установить $N = \prod_{i=1}^{b} p_i$. Для 1024-битной системы не желательно использование *b>3* (т. е. *N = pqr*) по соображениям безопасности, обсуждаемым ниже;

2) Выбирается тот же *e*, который используется в стандартных открытых ключах RSA. Затем вычисляется $d = e^{-1} mod\ \varphi(N)$. И, как и в классическом алгоритме, необходимо убедиться, что *e* взаимно простое с $\varphi(N) = \prod_{i=1}^{b}(p_i - 1)$.

Кодирование производится по тому же алгоритму, как и в классическом RSA. А декодирование выполняется с использованием китайской теоремы об остатках (CRT) [2]. Пусть $r_i = d\ mod(\ p_i - 1)$. Чтобы расшифровать зашифрованный текст *C*, сначала вычисляется $M_i = C^{r_i} mod\ p_i$ для каждого *i, 1 ≤ i ≤ b*. После чего уравнения объединяются в систему и применяется CRT, чтобы получить $M = C^d mod\ N$. Использование китайской теоремы об остатках занимает незначительное время по сравнению с возведением в степень.

Сравним работу по расшифровке по вышеописанной схеме с работой, проделанной при расшифровке текста, зашифрованного классическим RSA. Важно, что стандартное дешифрование RSA с использованием CRT требует два полных возведения в степень по модулю $\left[\frac{n}{2}\right]$ битных чисел. В расшифровке RSA с несколькими простыми числами требуется *b* возведений в степень по модулю $\left[\frac{n}{b}\right]$ битовых чисел. Используя стандартный алгоритм, вычисление $x^d mod\ p$ требует времени $O(\log d\ log^2\ p)$. Но в случае, когда *d* порядка *p*, время выполнения равно $O(log^3\ p)$. Следовательно, асимптотика ускорения RSA с несколькими простыми числами по сравнению со стандартным RSA равна:

$$\frac{2 \cdot (\frac{n}{2})^3}{b \cdot (\frac{n}{b})^3} = \frac{b^2}{4} \quad (1)$$

Для 1024-битного RSA мы можем использовать не более *b* = 3 (т. е. *N=pqr*), что дает теоретическое ускорение примерно в 2,25 раза по сравнению со стандартным дешифрованием RSA. На практике, эксперименты (реализованные с использованием языка Java) показывают, что среднее ускорение равно 1,73 раза по сравнению со стандартным RSA.

Безопасность данной модификации зависит от сложности факторизации целых чисел, которые образуют *N*. Для того, чтобы простые множители *N* было тяжело найти, нужно убедиться, что они не попадают в диапазон метода эллиптических кривых (ECM) [3]. В настоящее время 256-битные простые множители рассматриваются в рамках ECM, поскольку работа по нахождению такие факторы находятся в пределах объема работ, необходимых для факторингового проекта RSA-512. Следовательно, для 1024-битных модулей не следует использовать более трех множителей.

Можно еще больше ускорить расшифровку RSA, используя модуль вида $N = p^{b-1}q$, где *p* и *q* простые числа по $\left[\frac{n}{b}\right]$ бит каждый. Когда N имеет длину 1024 бита, мы можем использовать не более b = 3, т. е. $N = p^2q$, тогда простые числа *p, q* имеют длину 341 бит.

Алгоритм генерации ключа принимает в качестве входных данных параметр безопасности *N* и дополнительный параметр *b*. Он генерирует ключи RSA следующим образом:

1) Генерация 2 различных простых числа *p* и *q*, каждое длиной $\left[\frac{n}{b}\right]$ бит. Установить $N = p^{b-1}q$;

2) Выбирается тот же e, который используется в стандартных открытых ключах RSA. Затем вычисляется $d = e^{-1} mod\ (p-1)(q-1)$;

3) После чего вычисляется пара дополнительных секретных ключей $r_1 = d\ mod\ (p-1)$ и $r_2 = d\ mod\ (q-1)$;

Кодирование производится точно так же, как и в классическом RSA. А декодирование выполняется следующим образом:

В первую очередь рассчитываются $M_1 = C^{r1} mod\ p$ и $M_2 = C^{r2} mod\ q$, таким образом имеем ${M_1}^e = C\ mod\ p$ и ${M_2}^e = C\ mod\ q$. При помощи леммы Гензеля [4] можем вычислить такое $({M_1}')$, что $({M_1}')^e = C\ mod\ p^{b-1}$. Стоит так же отметить, что данная операция намного менее затратна, нежели возведение в степень (p^{b-1}). И последним шагом, как и в предыдущей модификации является использование CRT для поиска такого *M*, что $M = {M_1}'\ mod\ p^{b-1}$ и $M = M_2\ mod\ q$.

После чего, как и в RSA с несколькими простыми числами, можно получить исходное значение с помощью китайской теоремы об остатках.

Далее сравним время, необходимую для расшифровки с использованием RSA с несколькими степенями, с временем, которое требуется для стандартного RSA. Для RSA с несколькими степенями расшифровка требует двух возведений в степень по модулю $\left[\frac{n}{b}\right]$ битовых чисел и *b − 2* поднятия Гензеля (лемма Гензеля). Поскольку поднятие Гензеля происходит намного быстрее, чем возведение в степень, в расчёт берется только время для двух возведений в степень. Как отмечалось ранее, полное возведение в степень с использованием модульной арифметики, занимают кубическое время. Итак, ускорение данной модификации RSA по сравнению со стандартным RSA составляет примерно:

$$\frac{2 \cdot (\frac{n}{2})^3}{2 \cdot (\frac{n}{b})^3} = \frac{b^3}{8} \tag{2}$$

Для 1024-битного RSA b снова должно быть не более трех (т. е. $N = p^2q$), давая теоретическое ускорение в 3,38 раза по сравнению со стандартным расшифрованием RSA. Реальный опыт показывает, что на практике мы получаем среднее ускорение в 2,30 раза.

Безопасность RSA с несколькими степенями зависит от сложности разложения целых чисел на множители. Так же, как и в RSA с несколькими простыми числами, нужно убедиться, что простые множители N не совпадают и не подпадают попадают в диапазон ECM. Следовательно, для 1024-битных систем можно использовать не более b = 3, т. е. $N = p^2q$.

В результате работы были проанализированы и реализованы два варианта ускорения работы дешифрации алгоритмом RSA. Каждый из алгоритмов на практике показал увеличение производительности примерно в 2 раза. Так же, была проанализирована безопасность модификаций, из математическое модели были вынесены значение коэффициентов, которые обеспечивают увеличение производительности, не увеличивая уязвимость шифрования.

Список использованных источников:

1. Thermal behavior of the YAG precursor prepared by sol–gel combustion process / F. Qiu [et al.] // Ceramics International, 2005. – P. 663-665.

2. C Ding, D Pei and A Salomaa Chinese Remainder Theorem, Finland 1996

3. R. Silverman and S. Wagstaff Jr. “A Practical Analysis of the Elliptic Curve Factoring Algorithm.” Math. Comp. 61(203):445–462. Jul. 1993

4. H. Cohen. A Course in Computational Algebraic Number Theory, vol 138 of Graduate Texts in Mathematics. Springer-Verlag, 1996, p. 139