ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Эссе
Crystals-Dilithium

Бобров Александр
Группа Б01-904

1 Введение

Широко используемые на сегодняшний день ассиметричные системы шифрования основаны на двух типах задач теории чисел:

- факторизация целых чисел (RSA, схема Эль-Гамаля);
- дискретное логарифмирование (семейство алгоритмов Дефи-Хелмана).

Обращение этих задач считалось носуществивым за разумное время по причине отсутствия полиноминальных алгоритмов по времени выполнения. Но начиная с 1995-го года начинается период "квантовой революции"в теории алгоритмов.

В 1995 г. Питер Шор продемонстрировал полиномиальные алгоритмы обращения описанных выше задач на квантовых компьютерах [1]. В 1996 г. Гровер продемонстрировал общий метод поиска в базе дан- ных со сложностью $O(\sqrt{N})$, позволяющий реализовывать расшифровку симметричных алгоритмов шифрования эквивалентную двукратному у- меньшению ключа шифра [2].На практике работа алгоритма была провере- на на 2-х кубитном квантовом компьютере, состоящем из полумиллитра смеси изотопа карбона-13 помеченного хлороформом, находящегося в ацетоне-D6 [3].

Так, получается, что используемые на практике системы ассиметричного шифрования, как и основная стадия шифрования - хеширование перестали быть сложными для обращения. Как следствие, возникла потребность поиска задач и алгоритмов, основанных на этих задачах, решение и расшифрование которых не было бы возможно с помощью квантовых компьютеров.

2 SVP, NTRU

Такой задачей стала SVP (shortest vector problem). Это задача о нахождении кратчайшего вектора в дискретной целочисленной решётке, которая может быть представлена как множество векторов заданных целочисленными линейно независимыми базовыми векторами (все компоненты каждого вектора вычисляются по модулю некоторого целого числа). Основные преимущества этой задачи:

- возможность построить одностороннюю функцию с секретом, быстро обращаемую при наличии дополнительных сведений (trapdoor function);
- не разрешима за полиноминальное время даже на квантовых вычислителях;
- SVP является NP-полной задачей.

Первое свойство позволяет значительно ускорить генерацию ключей и вычисление подписей на решётках (скорость генерации ключа и подписи(проверки подписи) улучшается с $O(n^2)$ до $O(n)$). Второе и третье свойства гарантируют сложность взлома алгоритмов, построенных SVP.

Разновидностью SVP является CVP (closest vector problem). Это задача о нахождении вектора в решётке, ближайшего к выбранному (Найти вектор $\boldsymbol{x}$ такой, что он даёт минимальное расстояние до выбранного вектора $\boldsymbol{v}$ в выбранно решётке $\boldsymbol{L}$). Алгоритм Crystalls-Dilithium использует в своей основе как раз CVP. Кроме того, в этом алгоритме используется алгоритм, аналогичный NTRU.

NTRU - это система шифрования, основанная на задаче NTRU-свёртки модулярных решёток, которая является частным случаем CVP-задачи (итог - частный случай частного случая). Основой шифрования является опе- рация свертки на кольце модулярных многочленов (с целыми коэффициен- тами). Под сверткой многочленов в данном случае понимают, их умноже- нием, с заданным правилом свертки $x^i = 1$, где $i = const$. Например, $x^5 = 1$: $(2x^4 - 3x^3 + 2)(4x^5 + 2x - 2) = -2x^4 - 6x^3 + 4x + 8$
Под модулярным многочленом $Z[x]/(x^n - 1)\ modk$, понимают многочлен $P_k(x) = b_{n-1}x^n - 1 + ... + b_1x + b_0$ коэффициенты которого являются остатком от деления, коэффициентов исходного многочлена
$P(x) = b_{n-1}x^n - 1 + ... + b_1x + b_0$ на k и принадлежащие некоторому промежутку: $c_1 \leq b_i \leq c_2$. Обратным многочленом $P_k(x)$ по модулю k явялется могчлен $P_k^{-1}(x) : P_k^{-1}(x) * P_k(x) = 1\ modk$.

Тогда процесс шифрования будет заключаться в:

- выборе простого n, показателя степени для правила свертки, малого и большого взаимно простых модулей q и p;
- выборе многочленов $f(x), g(x) \in R$ с «малыми коэффициентами»;
- вычислении обратных многочленов $F_q(x) = f^{-1}(x)\ modq$

Публичным ключом является многочлен $h(x) = g(x) * F_q\ modq$, приватным - многочлен $f(x)$.

Шифрование

Для шифрования текста, представляемого многочленом $m(x)\ modp$ выбирается "малый"многочлен $r(x)$. Тогда сообщение шифруется по формуле $e = p * r * h + m\ modq$.

Расшифровка

Для расшифровки вычисляется $a(x) = e(x)*f(x)\ modq$, коэффициента этого многочлена будут $A \leq a_i < A + q$. Тогда исходное сообщение восстанавливается по формуле $m(x) = F_p * a\ modp$.

3 Fiat-Shamir heuristic

Ещё один протокол, который нужен для построения алгоритма Crystalls-Dilithium - это протокол Фиата-Шамира с прерываниями. На примере доказательства знания дискретного логарифма какого-то числа покажем поэтапную работу протокола:

1. Алиса хочет доказать, что знает число x, которое является дискретным логарифмом у, $y = g^x(modn)$, также стороны заранее договариваются о выбранном простом числе q.

2. Алиса берёт случайное число из кольца по модулю q ($\mathbf{Z_q^*}$) и вычисляет $t = g^v$.

3.Алиса вычисляет хеш-функцию $c = \mathbf{H}(g, y, t)$.

4.Алиса вычисляет $r = v - cx\ \ mod\lambda(q)$, здесь $\lambda(q)$ - это количество простых чисел от 1 до q. Результатом является доказательство (ключ) - пара (t, r).

5. Теперь, имея эту пару, кто угодно, знающий x, может доказать знание Алисы, вычислив истинность выражения $t = g^r y^c$.

Знание хеш-функции, числа q и случайного числа v считаются открытыми, тогда проверку сможет осуществить любой, имеющий эти знания, знания о значении x и знания о паре-доказательстве, выданному Алисой.
Кроме этого, важно, чтобы значение хэщ-функции зависело от значения y, так как иначе атакующий может подобрать любое подходящее значение y, так чтобы узнать значение cx. На практике применяется раундовый (итеративный) протокол Фиата-Шамира: проводится несколько раундов (пункты 1-5), и если все арунды завершаются подтверждением, то знание считается доказанным.
В алгоритме Crystalls-Dilithium используется протокол Фиата-Шамира с прерываниями. Дело в том, что задача(svp), на которой основан этот алгоритм решается приближённым методом, поэтому невозможно всегда гарантировать нахождение правильного решения. Для таких случаев используется протокол Фиата-Шамира с отбрасываниями. Его идея заключается в том, чтобы дока-

зать истинность знания не во всех раундах проверки, а только в части из них (например, в $\frac{2}{3}$ всех раундов проверки). Кроме того, может случиться и так, что вычисления займут слишком много времени. Чтобы не тратить лишние ресурсы, используются отбрасывания некоторых раундов.

4 Crystals-Dilithium, упрощённая схема

Здесь будут рассмотрены три этапа урощённой схемы алгоритма Crystals-Dilithium - генерация ключей, подпись и верификация.

Ниже приведена схема упрощённого алгоритма, а ещё ниже приведено подробное объяснение каждого шага алгоритма.

<u>Gen</u>

$\mathbf{A} \leftarrow R_q^{k\times\ell}$

$(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^\ell \times S_\eta^k$

$\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$

$\textbf{return } (pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$

<u>$\mathsf{Sign}(sk, M)$</u>

$\mathbf{z} := \perp$

$\textbf{while } \mathbf{z} = \perp \text{ do}$

$\quad \mathbf{y} \leftarrow S_{\gamma_1 - 1}^\ell$

$\quad \mathbf{w}_1 := \mathsf{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$

$\quad c \in B_\tau := \mathrm{H}(M \parallel \mathbf{w}_1)$

$\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$

$\quad \textbf{if } \|\mathbf{z}\|_\infty \geq \gamma_1 - \beta \text{ or } \|\mathsf{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty \geq \gamma_2 - \beta, \textbf{ then } \mathbf{z} := \perp$

$\textbf{return } \sigma = (\mathbf{z}, c)$

<u>$\mathsf{Verify}(pk, M, \sigma = (\mathbf{z}, c))$</u>

$\mathbf{w}_1' := \mathsf{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$

$\textbf{if return } [\![\|\mathbf{z}\|_\infty < \gamma_1 - \beta]\!] \textbf{ and } [\![c = \mathrm{H}(M \parallel \mathbf{w}_1')]\!]$

- Генерация ключей происходит в 4 этапа:
  - генерация случайной матрицы $\mathbf{A}$ размера $k \times l$. Эта матрица состоит из полиномов в кольце $R_q = \mathbf{Z_q}[X]/(X^n + 1)$;
  - генерация двух случайных секретных ключей - векторов $\mathbf{s_1}$ и $\mathbf{s_2}$. Каждый коэффициент этих векторов (множитель перед базисным вектором) - это элемент R_q, не больший заранее выбранного η. Размерность $\mathbf{s_1}$ - l, размерность $\mathbf{s_2}$ - k;
  - на следующем шаге генерируется вторая часть открытого ключа как $\mathbf{t} = \mathbf{A}\mathbf{s_1} + \mathbf{s_2}$;

– открытым ключом является набор $(\mathbf{A}, \mathbf{t})$, закрытым - набор $(\mathbf{A}, \mathbf{t}, \mathbf{s_1}, \mathbf{s_2})$.

- Подпись:

  – алгоритм подписания генерирует вектор-маску из полиномов с коэффициентами меньше, чем γ_1. Параметр γ_1 выбран так, что он достаточно большой, чтобы не раскрыть секретный ключ(алгоритм с нулевым знанием или zero-knowledge algorithm), но достаточно маленький, чтобы подпись нелегко было подделать;

  – после этого подписывающий вычисляет $\mathbf{Ay}$ и "биты высших порядков"(самые старшие биты этого произведения) записываются в $\mathbf{w_1}$. В самом деле, каждый коэффициет w в $\mathbf{Ay}$ может быть записан как $w = w1 * 2\gamma_2 + w_0$, где $|w_0| \leq \gamma_2$. Тогда интуитивно понятно, что $\mathbf{w}$ - это вектор, собирающий в себе все w_1;

  – тогда "испытание"c создаётся как хэш исходного сообщения и $\mathbf{w}$. Результатом будет многочлен в R_q с кожффициентами равными ± 1 или 0, причём количество $\pm$ обозначим как τ (на будущее). Это сделано для того, чтобы c имеет малую норму и размером от 128 до 256;

  – после этого потенциальная подпись вычисляется как
  $\mathbf{z} = \mathbf{y} + c\mathbf{s_1}$;

  Если бы $\mathbf{z}$ сразу выводился, до схема подписи была бы небезопасной, так как в этом случае происходила бы утечка секретного ключа. Чтобы избежать зависимости $\mathbf{z}$ от секретного ключа, мы используем подбор с отказом (как в разобранном протоколе Фиата-Шамира). Требуется обозначить условие условие, когда мы отбрасываем подпись и вычисляем новую.

  Пусть параметр β - это максимально возможный коэффициет в $c\mathbf{s_i}$. Так как c содержит в себе ровно τ 1 и -1, то $\beta \leq \tau * \eta$. Если какой-то коэффициент $\mathbf{z}$ больше, чем $\gamma_1 - \beta$, то процедура подписи начинается заново. Кроме этого условия, есть ещё одно: если какой-то коэффициент младших битов $\mathbf{Az} - c\mathbf{t}$ больше, чем $\gamma_2 - \beta$.

  Первое условие важно только для безопасности подписи, тогда как второе - и для безопасности, и для правильности алгоритма. Параметры кольца q и n позволяют добиться правильной подписи за небольшое число итераций (примерно 4 итерации для $q = s^3 3 - 2^1 3 + 1, n = 256$).

- Проверка подписи:

  – проверющий вычисляет вектор $\mathbf{w_1}'$ - вектор старших битов от $\mathbf{Az} - c\mathbf{t}$ и подтверждает подпись, если все коэффиуиенты $\mathbf{z}$ меньше, чем $\gamma_1 - \beta$ и

c - это результат хэш-функции сообщения и $\mathbf{w_1'}$

5 Улучшения упрощённой схемы

Самое заметное (но легко исправляемое) улучшение - это замена матрицы $r \times l$, состоящей из многочленов, в публичном ключе, так как её представление занимает очень много места в памяти. Решение: заменить матрицу $\mathbf{A}$ на семя ρ, с помощью которого алгоритм SHAKE-128 генерирует нужную нам матрицу. Тогда открытый ключ - это набор $(\rho, \mathbf{t})$, и его размер продиктован, в основном, размером $\mathbf{t}$.

Кроме этого, Dilithium уменьшает размер битового представления $\mathbf{t}$ немного больше, чем в два раза ценой увеличения подписи почти на сто байтов. При подтверждении подписи, $\mathbf{w_1'}$ не сильно зависит от младших битов $\mathbf{t}$, потому что $\mathbf{t}$ умножен на на очень "незначительный"(малые весса у коэффициентов) многочлен c. В приведённой схеме некоторые младшие биты $\mathbf{t}$ не включены в публичный ключ, и проверяющий не может всегда правльно посчитать старише биты $\mathbf{Az} - c\mathbf{t}$. Для этого подписывающий добавляет "подсказки"как часть подписи, которые существенно помогают, добавляя в произведение с c недостающие младшие биты $\mathbf{t}$. С этой подпаской можно правильно посчитать $\mathbf{w_1'}$.

Также можно рассмотреть улучшение общего случая умножения матрицы $\mathbf{A}$. Её элементы - полиномы в $\mathbf{Z_q}[X]/(X^{256}+1)$, которые умножаются на вектор $\mathbf{y}$ таких же многочленов. Как и во многих алгоритмах, основанных на решётках, кольцо можно выбрать так, чтобы умножение производилось очень эффективной операцией с помощью дискретного преобразовани Фурье. Для этого нужно выбрать простое q так, чтобы группа $\mathbf{Z_q^*}$ обладала элементом порядка $2n = 512$, или (что то же самое) $q = 1 \ mod 512$. Если r такой элемент, то $X^{256} + 1 = (X - r)(X_r^3)...(X - r^5 11)$ и следовательно, возможно эквивалентное представление любого полинома $a \in \mathbf{Z_q}[X]/(X^{256}+1)$ с помощью китайской теоремы об остатках в форме $(a(r), a(r^3), ..., a(r^{2n-1}))$. Преимущество этого представления в том, что произведение двух полиномов происходит покоординатно (как в скалярном произведении двух векторов). Таким образом, самая дорогая часть перемножения многочленов - это преобразование $a \to \hat{a}$ и обратное преобразование $\hat{a} \to a$, а это известные и быстрые дискретные преобразования Фурье.

Источники

[1] Shor P.W., *Polynomial-time algorithms for prime factorization and dis- crete logarithms on a quantum computer*. SIAM J. Com.,1997, 26:5, стр. 1484-1509.

[2] Grover L. K., *A fast quantum mechanical algorithm for database search, Proceedings of the 28th ACM STOC, 1996, стр. 212–219.*

[3] Chuang I. L., Gershenfeld N., Kubinec M., *Experimental Implementation of Fast Quantum Searching, Physical Review Letters, 1998, 80:15, стр. 3408–3411.*