Теория чисел в криптографии

Владимир Борисенко

Мехмат МГУ

vladibor2016@yandex.ru
vladimir_borisen@mail.ru

Классическое и блочное шифрование

- При **блочном шифровании** файл с информацией представляется последовательностью нулей и единиц. Последовательность делится на блоки фиксированной длины, каждый блок шифруется и передается по отдельности.
- Каждый блок можно рассматривать как двоичную запись целого числа. Шифрование — это функция, отображающая исходное целое число на зашифрованное:

$$E : \mathbb{Z} \to \mathbb{Z}$$

  Обратная функция осуществляет расшифровку:

$$D : \mathbb{Z} \to \mathbb{Z}, \quad \forall x \in \mathbb{Z} \quad D(E(x)) = x.$$

- Поскольку размер блока ограничен, можно рассматривать не произвольные целые числа, а элементы кольца вычетов $\mathbb{Z}_m$.

Кольцо вычетов по модулю m

- Пусть $m \in \mathbb{Z}$ — ненулевое целое число. Кольцо вычетов $\mathbb{Z}_m$ — это фактор кольцо кольца $\mathbb{Z}$ по идеалу, порожденному элементом m. Оно состоит из классов эквивалентности. Два числа $x, y \in \mathbb{Z}$ эквивалентны, если $m \mid (x - y)$.
- Число классов эквивалентности равно m. Операции над классами определяются через из представителей.
- Если в каждом классе выбрать по представителю, то получим *систему остатков*, представляющую кольцо $\mathbb{Z}_m$. Например,

$$\begin{array}{rcl} \mathbb{Z}_5 & = & \{0,1,2,3,4\} \quad \text{или} \\ \mathbb{Z}_5 & = & \{-2,-1,0,1,2\} \end{array}$$

- Числа $x, y \in \mathbb{Z}$, попадающие в один класс кольца $\mathbb{Z}_m$, называются *сравнимыми по модулю m*:

$$x \equiv y \pmod{m}$$

Например,

$$-1 \equiv 4 \pmod{5}$$

Малая теорема Ферма

Самая замечательная теорема в теории чисел — это

Малая теорема Ферма. Пусть $p \in \mathbb{Z}$ — простое число. Тогда для любого $b \in \mathbb{Z}$, $b \not\equiv 0 \pmod{p}$ справедливо сравнение

$$b^{p-1} \equiv 1 \pmod{p}.$$

Другая ее форма:
Пусть $p \in \mathbb{Z}$ — простое число. Тогда для любого $b \in \mathbb{Z}$ и для любого $k \in Z$, $k \geq 0$ справедливо сравнение

$$b^{k(p-1)+1} \equiv b \pmod{p}.$$

Базовые алгоритмы в теории чисел

- Алгоритм Евклида вычисляет наибольший общий делитель двух чисел.
$$d = \gcd(m, n)$$
- Расширенный алгоритм Евклида вычисляет наибольший общий делитель двух чисел и его представление в виде линейной комбинации исходных чисел
$$\begin{array}{rcl} d & = & \gcd(m, n), \\ d & = & u \cdot m + v \cdot n. \end{array}$$
- Расширенный алгоритм Евклида позволяет вычислить обратный к n элемент в кольце $\mathbb{Z}_m$. Если $gcd(m, n) = 1$, то
$$1 = u \cdot m + v \cdot n \equiv v \cdot n \pmod{m},$$
то есть v — обратный к n элемент в $\mathbb{Z}_m$.

- алгоритм *быстрого возведения в степень* чрезвычайно часто используется в криптографии и других приложениях теории чисел, поскольку позволяет вычислить степень a^n, выполнив всего лишь $O(\log_2 n)$ арифметических операций.

Реализация алгоритма Евклида на языке Python3:

```
def gcd(m, n):
    while n != 0:
        (m, n) = (n, m%n)
    if m >= 0:
        return m
    else:
        return (-m)
```

В алгоритме Евклида в цикле пара (m, n) заменяется на пару (n, r), где r — остаток от деления m на r. При этом множества общих делителей пар (m, n) и (n, r) совпадают, т.е. величина НОД(m, n) является *инвариантом цикла*. Цикл заканчивается, когда $n = 0$, при этом НОД$(m, 0) = m$.

Расширенный алгоритм Евклида:

```
def extEucl(m, n):
    (a, b) = (m, n)
    u1 = 1; v1 = 0;
    u2 = 0; v2 = 1
    while b != 0:
        assert (a == u1*m + v1*n and b == u2*m + v2*n)
        q = a // b; r = a % b
        assert (r == a - q*b)
        (a, b) = (b, r)
        (u1, u2) = (u2, u1 - q*u2)
        (v1, v2) = (v2, v1 - q*v2)
    if a >= 0:
        return (a, u1, v1)
    else:
        return (-a, -u1, -v1)
```

Выполняется обычный алгоритм Евклида: $(a, b) \to (b, r)$, при этом хранится выражение чисел a, b в виде линейных комбинаций исходных чисел m, n: $a = u_1 m + v_1 n$, $b = u_2 m + v_2 n$.

Алгоритм быстрого возведения в степень в кольце $\mathbb{Z}_m$:

```
def powermod(a, n, m):
    assert(n >= 0)
    p = 1
    while n > 0:
        # Invariant: p*a^n
        if n%2 == 0:
            n //= 2
            a = (a*a)%m
        else:
            n -= 1
            p = (p*a)%m
    return p
```

Инвариантом цикла является величина $p \cdot a^n$, начальное значение $p = 1$, при этом n в цикле убывает. По завершению $n = 0$, значит, ответ содержится в переменной p.

Китайская теорема об остатках

Китайская теорема об остатках позволяет свести изучение колец $\mathbb{Z}_m$ для произвольных $m \in \mathbb{Z}$ к примарным кольцам вида $\mathbb{Z}_{p^s}$, где p — простое число.

Теорема. Пусть $m = m_1 m_2 \dots m_k$, и $\gcd(m_i, m_j) = 1$ для $i \neq j$. Тогда

$$\mathbb{Z}_m \cong \mathbb{Z}_{m_1} \oplus \mathbb{Z}_{m_2} \oplus \dots \oplus \mathbb{Z}_{m_k}.$$

В частности, если $m = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, то

$$\mathbb{Z}_m \cong \mathbb{Z}_{p_1^{e_1}} \oplus \mathbb{Z}_{p_2^{e_2}} \oplus \dots \oplus \mathbb{Z}_{p_k^{e_k}}.$$

Например, $\mathbb{Z}_{105} \cong \mathbb{Z}_3 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_7$. Элементами прямой суммы являются строки:

$$34 \cong (1, -1, -1)$$

Функция Эйлера

Функцией Эйлера $\phi(m)$ называется число обратимых элементов кольца $\mathbb{Z}_m$. Поскольку в этом кольце обратимы элементы, взаимно простые с m, то в эквивалентной формулировке функция Эйлера $\phi(m)$ равна числу элементов x таких, что $0 < x < m$ и $\gcd(x, m) = 1$.

Китайская теорема об остатках позволяет вычислить функцию Эйлера: если $m = m_1 m_2 \dots m_k$, $\gcd(m_i, m_j) = 1$ для $i \neq j$, то

$$\phi(m) = \phi(m_1)\phi(m_2)\dots\phi(m_k).$$

Для примарного кольца

$$\phi(p^s) = p^s - p^s/p = (p-1)p^{s-1}.$$

Окончательно получаем для $m = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

$$\phi(m) = (p_1 - 1)p_1^{e_1 - 1}(p_2 - 1)p_2^{e_2 - 1}\dots(p_k - 1)p_k^{e_k - 1}$$

В частности, при $m = pq$, где p, q — простые,

$$\phi(m) = (p-1)(q-1).$$

Теорема Эйлера

Теорема Эйлера обобщает Малую теорему Ферма.
Теорема Эйлера. Пусть элемент $b \in \mathbb{Z}_m$ обратим, т.е. $\gcd(b, m) = 1$. Тогда
$$b^{\phi(m)} \equiv 1 \pmod{m}.$$

Нам понадобится частный случай теоремы Эйлера для числа m, свободного от квадратов.
Частный случай теоремы Эйлера. Пусть число $0 \neq m \in Z$ свободно от квадратов, т.е.
$$m = p_1 p_2 \dots p_k,$$
где p_i — простые числа. Тогда для любого $b \in \mathbb{Z}_m$ и любого $s \in \mathbb{Z}$, $s \geq 0$ справедливо сравнение
$$b^{s \cdot \phi(m)+1} \equiv b \pmod{m}.$$

Схема кодирования с открытым ключом RSA

Названа по имени ее авторов — Р.Ривест, А.Шамир, Л.Адлеман (1977). Процедуры шифрования и дешифровки в ней взаимно обратны, и одна не восстанавливается по другой.
Пусть $m \in \mathbb{Z}$ есть произведение двух больших простых чисел:

$$m = pq, \quad p, q \text{ — простые.}$$

Число m открыто, но его разложение на множители секретно. Выбирается произвольное число $e \in \mathbb{Z}_{\phi(m)}$, взаимно простое с $\phi(m)$; через $\phi(m)$ обозначается *функция Эйлера*. С помощью расширеного алгоритма Евклида вычисляется его обратный элемент $d \in \mathbb{Z}_{\phi(m)}$:

$$\begin{aligned} & e \cdot d \equiv 1 \pmod{\phi(m)}, \text{ где} \\ & \phi(m) = (p-1)(q-1) \ \text{ — функция Эйлера.} \end{aligned}$$

Число e открыто, число d секретно. Пара (m, e) представляет собой *открытый ключ*, пара (m, d) — *секретный ключ*.

Шифрующая процедура E (*Encryption*):

$$E : \mathbb{Z}_m \to \mathbb{Z}_m, \quad E(x) \equiv x^e \pmod{m},$$

Процедура дешифровки D (*Decryption*):

$$D : \mathbb{Z}_m \to \mathbb{Z}_m, \quad D(y) \equiv y^d \pmod{m},$$

Эти две процедуры взаимно обратны:

$$\forall x \in \mathbb{Z}_m \quad D(E(x)) = x, \quad \forall y \in \mathbb{Z}_m \quad E(D(y)) = y.$$

Доказательство этого факта вытекает из теоремы Эйлера. Поскольку $ed \equiv 1 \pmod{\phi(m)}$, то $ed = 1 + k\phi(m)$ для некоторого $k \in \mathbb{Z}$, $k \geq 0$.

$$D(E(x)) \equiv D(x^e) \equiv x^{ed} \equiv x^{1+k\phi(m)} \equiv x \pmod{m}.$$

Генерация и взлом ключей в схеме RSA

Для *генерации двух ключей* — открытого и секретного — в схеме RSA нужно уметь генерировать большие простые числа длиной не меньше 150 десятичных цифр.

Для *взлома ключа* надо по открытому ключу (m, e) вычислить число d такое, что $ed \equiv 1 \pmod{\phi(m)}$, а для этого надо знать функцию Эйлера $\phi(m) = (p-1)(q-1)$. Задача вычисления функции Эйлера $\phi(m)$ эквивалентна задаче разложения числа m на множители — *факторизации* числа m.

В настоящее время мы умеем генерировать простые числа большой длины, но для задачи факторизации не существует алгоритма, позволяющего за разумное время разложить на множители, например, 300-значное десятичное число. Стойкость криптографической схемы RSA основана на трудности задачи факторизации. Если будет реализован *квантовый компьютер*, для которого имеется *алгоритм целочисленной факторизации Шора*, то схему RSA больше нельзя будет использовать.

Тест Ферма и кармайкловы числа

Если m — простое число и b не делится на m, то по Малой теореме Ферма
$$b^{m-1} \equiv 1 \pmod{m}.$$
Отсюда вытекает, что, если для какого-либо $b \in \mathbb{Z}$, $2 \leq b < m-1$ выполняется неравенство
$$b^{m-1} \not\equiv 1 \pmod{m},$$
то m — составное число.
Вычисление $b^{m-1} \pmod{m}$ называется **Тестом Ферма для основания** b. Он позволяет получить доказательство того, что число m составное.
Подавляющее большинство составных чисел выявляется с помощью теста Ферма, однако, увы, не все. Первое такое число 341 было найдено французским математиком П.Саррусом в 1819 г.:
$$2^{340} \equiv 1 \pmod{341}, \quad 341 = 31 \cdot 11.$$

Кармайкловы числа

Число 341 не удовлетворяет тесту Ферма для основания 3:

$$3^{340} \equiv 56 \pmod{341}.$$

Однако есть такие парадоксальные числа, которые ведут себя как простые в Малой теореме Ферма для любых оснований b, взаимно простых с m, — это *кармайкловы числа*, названные так в честь американского математика Р.Кармайкла, открывшего первое такое число 561 в 1910 г.

Определение. Число $m \in \mathbb{Z}$ называется *кармайкловым*, если m составное и для всякого $b \in \mathbb{Z}$, взаимно простого с m, выполняется заключение Малой теоремы Ферма:

$$b^{m-1} \equiv 1 \pmod{m}.$$

Минимальные кармайкловы числа — это $561, 1105, 1729, \ldots$. Совсем недавно доказано, что множество кармайкловых чисел бесконечно. Тест Ферма не годится в качестве теста простоты для кармайкловых чисел.

Тест простоты Миллера–Рабина

Но тест Ферма можно немного исправить, чтобы он отсеивал и Кармайкловы числа. Такой вероятностный тест был предложен в конце 1970-х в совместных работах нескольких математиков.

Вероятностный тест простоты Миллера–Рабина. Проверяем на простоту нечетное число $m \in \mathbb{Z}$. Пусть

$$m - 1 = t \cdot 2^s, \text{ где } t \text{ — нечетное число.}$$

Выбираем случайное основание $b \in \mathbb{Z}$, $2 \leq b < m - 1$, и вычисляем последовательность

$$x_0 \equiv b^t, \; x_1 \equiv x_0^2, \; x_2 \equiv x_1^2, \; \dots, \; x_s \equiv x_{s-1}^2 \equiv b^{m-1} \pmod{m}.$$

В последовательности $x_0, x_1, x_2, \dots, x_s$ каждый следующий член является квадратом предыдущего. Если

(1) последний элемент последовательности отличен от 1 или

(2) в последовательности есть фрагмент $x_i \not\equiv \pm 1$, $x_{i+1} \equiv 1 \pmod{m}$,

то тест выдает ответ "m — составное число". Иначе тест выдает ответ "не знаю".

Тест простоты Миллера–Рабина

Теорема. Если тест Миллера–Рабина выдает ответ "m — составное число", то m действительно составное. Если же тест выдает ответ "не знаю", то вероятность того, что число m составное, не превышает $1/4$.

Пример применения теста Ферма. Покажем, что третье кармайклово число $m = 1729$ не простое. Выберем случайное основание $b = 123$. Имеем

$$m - 1 = 1728 = 27 \cdot 2^6.$$

Вычисляем последовательность

$$123^{27} \equiv 1464, \ 1464^2 \equiv 1065, \ 1065^2 \equiv 1.$$

Значит, m составное.

Алгоритмы факторизации Полларда

Первый алгоритм Полларда — ρ-алгоритм — основан на поиске цикла в рекуррентной последовательности. Пусть надо разложать на множители число m. Рассмотрим случайное начальное число $x_0 \in \mathbb{Z}_m$ и вычисляем бесконечную рекуррентную последовательность

$$x_0, \; x_1 = f(x_0), \; x_2 = f(x_1), \; x_3 = f(x_2), \ldots,$$

где $f(x) \equiv x^2 + 1 \pmod{m}$. Параллельно вычисляем

$$\gcd(x_0 - x_1, m), \; \gcd(x_1 - x_3, m), \; \gcd(x_2 - x_5, m), \ldots$$

На i-м шаге вычисляется $\gcd(x_i - x_{2i+1}, m)$. Алгоритм заканчивается, как только наибольший общий делитель станет отличен от 1. Работа алгоритма основана на том, что, если $p|m$, то последовательность $x_0, x_1, x_2, \ldots$ является циклической по модулю p, и рано или поздно будет выполнено сравнение

$$x_i \equiv x_j \pmod{p} \; \Rightarrow p \mid \gcd(x_i - x_j, m).$$

ρ-алгоритм факторизации Полларда на псевдокоде

```
алгоритм rhoFactorization(m, maxSteps = 1000000)
| x = случайное целое число в диапазоне 2 <= x < m-1
| y = x*x + 1 (mod m)
| d = gcd(x - y, m)
| steps = 0
|
| цикл пока d == 1 и d != m и steps < maxSteps:
| | x = x*x + 1 (mod m)    # отображение f: x -> x^2 + 1
| | y = y*y + 1 (mod m)    # применяется один раз к элементу x
| | y = y*y + 1 (mod m)    # и дважды к элементу y
| | d = gcd(x - y, m)
| | steps = steps + 1
| конец цикла
|
| если d == m   # Неудача...
| | то d = 1
| конец если
|
| вернуть d
конец алгоритма
```

$P-1$ алгоритм Полларда

Пусть нужно разложить на множители число m и пусть $p|m$ — простой делитель числа m. Предположим, что число $p-1$ гладкое (smooth) — число называется *N-гладким*. если оно раскладывается в произведение не очень больших степеней простых чисел:

$$p-1=q_1^{\alpha_1}q_2^{\alpha_2}\dots q_k^{\alpha_k},\ \text{причем}\ q_i^{\alpha_i}\le N\ \text{для всех}\ i=1,\dots,k.$$

Выберем случайное основание $b_0\in\mathbb{Z}$, $2\le b<m-1$, и рассмотрим последовательность

$$b_0,\ \ b_1\equiv b_0^2,\ \ b_2\equiv b_1^3,\ \ \dots,\ \ b_{N-1}\equiv b_{N-2}^N \pmod m.$$

Из гладкости числа $p-1$ следует, что $(p-1)\mid N!$, т.е. $N!=(p-1)\cdot k$. Значит, по малой теореме Ферма

$$b_{N-1}\equiv b_0^{N!}\equiv b_0^{(p-1)k}\equiv 1 \pmod p.$$

Следовательно, $p\mid \gcd(b_{N-1}-1,m)$ и мы нашли нетривиальный делитель числа m. Наибольший общий делитель $d=\gcd(b_i-1,m)$ вычисляется на каждой итерации цикла, поскольку мы заранее не знаем степени гладкости числа $p-1$.

$p-1$-алгоритм факторизации Полларда на псевдокоде

```
алгоритм p1Factorization(m, upperBound = 1000000)
| b = случайное число в интервале 2 <= b < m - 1
| d = gcd(b - 1, m)
| s = 2
|
| цикл пока d == 1 и s <= upperBound
| | b = b^s (mod m)
| | d = gcd(b - 1, m)
| | s = s + 1
| конец цикла
|
| если d == m   # Неудача...
| | d = 1
| конец если
|
| вернуть d
конец алгоритма
```