

Teoría de Números

$a \equiv b \pmod{n}$ si $a = b + kn$ para algún entero k . b se llama el **resíduo** de a , módulo n . a es **congruente a** b , módulo n . Los enteros $0 \dots n - 1$ forman el **conjunto completo de resíduos** módulo n . (*Nota:* la función de librería $\text{mod}()$, o el operador $\%$ en el lenguaje C, no siempre siguen la definición precisa.)

Aritmética modular es conmutativa, asociativa y distributiva.

Además:

$$(a + b) \pmod{n} = ((a \pmod{n}) + (b \pmod{n})) \pmod{n}$$

$$(a - b) \pmod{n} = ((a \pmod{n}) - (b \pmod{n})) \pmod{n}$$

$$ab \pmod{n} = ((a \pmod{n})(b \pmod{n})) \pmod{n}$$

$$(a(b + c)) \pmod{n} = ((ab \pmod{n}) + (ac \pmod{n})) \pmod{n}$$

Una operación criptográfica común es la **exponenciación modular**, $a^x \pmod{n}$, ej.:

$$a^9 \pmod{n} = (((a^2) \pmod{n})^2 \pmod{n})^2 \pmod{n} a \pmod{n}$$

Números Primos

Un número primo es un entero mayor que 1 cuyos únicos factores son 1 y él mismo. Hay una cantidad infinita de primos. Los de interés para la criptografía son números grandes (512 bits o más).

a y n son **relativamente primos** si no tienen factores en común sino “1”, o sea que $\gcd(a, n) = 1$. Calcular $\gcd(a, n)$ es fácil usando el conocido Algoritmo de Euclídes.

a y b son **inversos multiplicativos** si $ab \bmod n = 1$ (o $ab \equiv 1 \pmod{n}$). El inverso de a se escribe a^{-1} . No siempre existe.

En general: si a y n no son relativamente primos, entonces $a^{-1} \equiv x \pmod{n}$ no tiene solución. Si lo son, puede haber más de una solución. Para garantizar una solución única, basta que n sea primo. En general, encontrar un inverso es más costoso que gcd, pero todavía es tratable.

Misceláneos ...

- Los números $n \bmod p$, cuando p es primo o una potencia de un primo forman un **cuerpo finito**, también llamado un **cuerpo de Galois**, y denotado $\mathbf{GF}(p)$. Están definidos los operadores de suma, resta, multiplicación, y división por un factor distinto a 0. Existe una identidad aditiva (0) y otra multiplicativa (1). Cada número distinto a 0 tiene inverso único. Muchos criptosistemas se basan en $\mathbf{GF}(p)$, donde p es un primo grande.
- *Teorema:* si m es primo, y no es un factor de a , entonces

$$a^{m-1} \equiv 1 \pmod{m}$$

Esto se conoce como “El Pequeño Teorema de Fermat”.

- *Definición:* el **conjunto reducido de residuos** $\bmod n$ es el subconjunto de residuos que sean relativamente primos a n . Si n es primo, es el conjunto $1, 2, \dots, n - 1$. La **función indicatriz de Euler** (*Euler's totient function*), $\phi(n)$, es el número de elementos en este conjunto.

Si $\gcd(a, n) = 1$, $a^{\phi(n)} \bmod n = 1$.

Si n es primo, $\phi(n) = n - 1$. Si $n = pq$, donde p y q son primos, entonces $\phi(n) = (p - 1)(q - 1)$.

Factorización

El problema de la **factorización** es el de hallar los factores primos de un entero n . Un resultado básico es que tal factorización es única para cada n . Esto se conoce como *El Teorema Fundamental de la Aritmética*.

La complejidad de la factorización es mucho mayor que la de la multiplicación. El método ingenuo es “probar todos los primos menores que \sqrt{n} ”, pero el número de posibles candidatos es aproximadamente $\frac{\sqrt{n}}{\ln \sqrt{n}}$.

Sin embargo, el área es extremadamente activo, y varios algoritmos novedosos se han inventado en los últimos años:

- **Criba Cuadrática** (*Quadratic Sieve* o QS): el mejor método para números de menos que 110 dígitos (decimales). Las mejores versiones tienen complejidad temporal asintótica de $e^{(1+O(1))(\ln n)^{\frac{1}{2}}(\ln \ln n)^{\frac{1}{2}}}$
- **Criba de Cuerpo Numérico** (*Number Field Sieve* o NFS): el mejor conocido para más de 110 dígitos. La complejidad asintótica se estima en $e^{(1.923+O(1))(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}}$

Algunos Números ...

- En 1970, se logró factorizar un número “difícil” de 41 dígitos (*difícil* quiere decir que no tiene factores pequeños).
- En 1993 se factorizó un número difícil de 120 dígitos usando QS. El cálculo tardó tres meses y usó 825 Mips-años de poder de cómputo.
- En 1994, una variante del QS se usó para factorizar RSA-129 (un “reto” de RSA Data Security, Inc., de 129 dígitos). Lo hizo un equipo de 600 personas y 1600 máquinas en el Internet (incluyendo recursos del LDC/USB), estimado en 4000 a 6000 Mips-años. Se piensa que el método NFS sería 10 veces más rápido.

Más Números ...

Los algoritmos asimétricos importantes usan número primos grandes. Algunas preguntas:

- Si todo el mundo necesita un número primo diferente, ¿hay suficientes?

Hay aproximadamente 10^{151} primos de ≤ 512 bits. Si cada átomo en el universo “necesitara” 10^9 primos nuevos cada μ segundo desde el comienzo del universo hasta ahora, sólo se necesitarían 10^{109} primos – casi 10^{151} sobran ...

- ¿Qué pasa si dos usuarios escogen el mismo primo, por accidente?

No va a pasar al menos que el método de generación sea defectuoso.

- Si alguien crea una base de datos de todos los primos, ¿no podría usarla para atacar al criptosistema?

Sí, pero si pudieramos guardar 1 Gb en un disco de 1 gramo de peso, la lista de primos de sólo 512 bits sería tan pesado que la base de datos se convertiría en un hueco negro ...

Generación de Primos

Si la factorización es difícil, ¿cómo encontramos primos aleatorios grandes? Resulta que la determinación de **primalidad** es mucho más fácil que la factorización.

Varios métodos probabilísticos se han inventado. Uno de los mejores es el de **Rabin y Miller**:

1. Escoger un candidato p . Calcular b , el número de veces que 2 divide a $p - 1$. Calcular m , tal que $p = 1 + 2^b m$
2. Escoger un número aleatorio, $a < p$
3. Inicializar $j \leftarrow 0$ y $z \leftarrow a^m \bmod p$
4. Si $z = 1$ o $z = p - 1$, entonces p *podría ser primo*
5. Si $j > 0$ y $z = 1$, p *no es primo*
6. $j \leftarrow j + 1$. Si $j < b$ y $z \neq p - 1$, entonces $z \leftarrow z^2 \bmod p$ repetir el paso anterior. En cambio, si $z = p - 1$, p *podría ser primo*
7. Si $j = b$ y $z \neq p - 1$, entonces p *no es primo*

Si la prueba no falla para un valor determinado de a , éste se llama un **testigo**. Se puede mostrar que la probabilidad de que p sea compuesto (no primo) es no mayor que $\frac{1}{4}$. Si repetimos con t testigos, esa probabilidad se reduce a $\frac{1}{4^t}$.

El Mundo Real

En la práctica, se hace esto:

1. Generar un p de n bits, al azar.
2. Forzar el primero y último bit de p a que sean 1.
3. Prueba divisiones con primos pequeños (ej. hasta 2000).
Simplemente probando con 3, 5 y 7 elimina 54% de los números impares. Siguiendo hasta 256 elimina 80%.
4. Realizar la prueba Rabin-Miller para 5 valores pequeños de a
5. Si p falla, escoja otro y empezar de nuevo.

A veces necesitamos $n = pq$, donde p y q son **primos fuertes**, es decir que n no es fácil factorizar. Algunas propiedades que se han recomendado en la literatura:

- Que $\gcd(p - 1, q - 1)$ sea pequeño
- Que $p - 1$ y $q - 1$ tengan factores primos grandes, p' y q'
- Que $p' - 1$ y $q' - 1$, $p + 1$ y $q + 1$ tengan factores primos grandes
- Que $(p - 1)/2$ y $(q - 1)/2$ sean primos (implica las primeras dos condiciones)

También hay argumentos en contra de estas tácticas (ej. los primos no son tan aleatorios).