# Modular Exponentiation

Repeated square-and-multiply alg for exponentiation in $\mathbb{Z}_n$

Input: $a \in \mathbb{Z}_n$, $0 \leq k < n$ $\longrightarrow$ binary representation $k = \sum_{i=0}^{t} k_i 2^i$

Output: $a^k \bmod n$

1. Set $b \leftarrow 1$. if $k=0$, Return $(b)$
2. Set $A \leftarrow a$     variable   Value   $b \leftarrow a$
3. if $k_0 = 1$ then
4. For $i = 1 \sim t$
   - 4.1 $A \leftarrow A^2 \pmod n$
   - 4.2 if $k_i = 1$ then $b \leftarrow A \cdot b \pmod n$
5. return $(b)$

---

$a^{26}$

$26 = 2 \times 13 + 0$

$13 = 2 \times 6 + 1$

$6 = 2 \times 3 + 0$

$3 = 2 \times 1 + 1$

$2 \times 0 + 1$

$26 = (11010)_2$

$k_4 \; k_3 \; k_2 \; k_1 \; k_0$

1. $b = 1$, $k = 26 \neq 0 \rightarrow$ NO
2. $A = a$
3. $k_0 = 0 \neq 1 \rightarrow$ NO
4. $i \sim 4$
   $1$

$\boxed{i=1}$ $A = A^2 \pmod n$

if $k_1 = 1$ ✓ $\longrightarrow$ $b = A^2 \cdot 1 = A^2 \pmod n$

$\boxed{i=2}$ $A = (A^2)^2 = A^4 \pmod n$

if $k_2 = 0 \neq 1$ X

$\boxed{i=3}$ $A = (A^4)^2 = A^8 \pmod n$

if $k_3 = 1$ ✓ $\longrightarrow$ $b = A^8 \cdot A^2 = A^{10} \pmod n$

$\boxed{i=4}$ $A = (A^8)^2 = A^{16} \pmod n$

if $k_4 = 1$ ✓ $\longrightarrow$ $b = A^{16} \cdot A^{10} = A^{26} \pmod n$

5. return $A^{26} \pmod n$

$a^7$

$7 = 2 \times 3 + 1$

$3 = 2 \times 1 + 1$

$1 = 2 \times 0 + 1$

$(111)_2$ — $k_2$ $k_1$ $k_0$

1. $b \leftarrow 1$ if $k = 0$ X
2. $A = a$
3. if $k_0 = 1 \longrightarrow b = a$
4. $i = 1 \sim 2$

$\boxed{i=1}$

$A = A^2 \pmod n$

if $k_1 = 1$ ✓ $\longrightarrow b = A^2 \cdot a = A^3 \pmod n$

$\boxed{i=2}$

$A = (A^2)^2 = A^4 \pmod n$

if $k_2 = 1$ ✓ $\longrightarrow b = A^4 \cdot A^3 = A^7 \pmod n$

5. return $(A^7 \pmod n)$

---

## $\boxed{\text{Fermat primality test alg.}}$

Input: an odd integer $n \geq 3$ & security parameter $t \geq 1$ → # of iteration of your loop/try

Output: answer $n$ is "prime" or "composite"

1. For $i = 1 \sim t$ do
   - 1.1 chose random integer $a$ $2 \leq a \leq n-2$ (previous odd number)
   - 1.2 Compute $r = a^{n-1} \pmod n$ using S-&-M algorithm
   - 1.3 If $r \neq 1$ then return "Composite"
2. Return("prime")

Fermat's Th

$$\boxed{a^{p-1} \equiv 1 \pmod p}$$

**1 Execution**

$n=25 \qquad t=3$

$i=1 \rightsquigarrow 3$

$\boxed{i=1}$ $\quad 2 \leqslant a=9 \leqslant 23$

Compute $r=9^{24} \pmod{25}=11$

if $r\neq 1$ ✓ then return "Composit"

**2 Execution**

$n=25 \qquad t=3$

$i=1 \longrightarrow 3$

$\boxed{i=1}$ $\quad 2 \leqslant a=7 \leqslant 23$

compute $r=7^{24} \pmod{25}=1$

if $r\neq 1$ X

$\boxed{i=2}$ $\quad 2 \leqslant a=4 \leqslant 23$

Compute $r=4^{24} \pmod{25}=6$

if $r\neq 1$ ✓ return "Composite"

**3 Execution**

$n=25 \qquad t=2$

$i=1 \longrightarrow 3$

$\boxed{i=1}$ $\quad 2 \leqslant a=7 \leqslant 23$

$r=7^{24} \pmod{25}=1$

if $r\neq 1$ X not true

$\boxed{i=2}$ $\quad 2 \leqslant a=18 \leqslant 23$

$r=18^{24} \pmod{25}=1$

if $r\neq 1$ X not true

return("prime")

Here, we set $t=2$ & after 2 iterations, the algorithm returns "prime" which is NOT correct because 25 is not prime. That is why it doesn't work properly all the time & it depends on your security parameter.

## $\boxed{\text{Miller\_Rabin probabilistic primality test}}$

Input: an odd integer $n \geq 3$ and $t \geq 1$ (security parameter)

Output: "Is $\underline{n}$ prime?"

1. write $\underbrace{n-1}_{\text{even}} = \underbrace{\underbrace{2^s}_{\text{even}} * \underbrace{r}_{\text{odd}}}_{\text{even}}$ s.t. "$r$" is odd $\longrightarrow$

$n = 25$

$n-1 = 24 = 2^3 \times 3$

$s = 3$, $r = 3$

| | |
|---|---|
| 24 | 2 |
| 12 | 2 |
| 6 | 2 |
| 3 | 3 |
| 1 | |

2. For $i = 1 \rightsquigarrow t$ do

Loop

- 2.1 Choose a random integer $a$, $2 \leq a \leq \overset{\text{previous odd number}}{n-2}$
- 2.2 Compute $y = a^r \pmod{n}$ using $\underline{\text{S\_8\_M algo.}}$
- 2.3 If $y \neq 1$ and $y \neq n-1$ do
  - $j = 1$
  - while $j \leq s-1$ and $y \neq n-1$ do

    Loop

    - Compute $y \leftarrow y^2 \pmod{n}$
    - if $y = 1$ then return "Composit"
    - $j \leftarrow j+1$
  - If $y \neq n-1$ then return "Composite"

3. Return "prime".