Euler Totient Function

$\phi(n)$: # of integers between 1 and n (inclusive) that is co-prime with n.

Examples. $\phi(26)=12$, $\phi(1000)=400$.

Claim: $\boxed{\phi(n) = n \cdot \prod_{\substack{p \mid n \\ p \text{ is a prime}}} (1-\frac{1}{p})}$.

Example: $1000 = 2^3 \cdot 5^3$ $\quad \phi(1000) = 1000 \cdot (1-\frac{1}{2}) \cdot (1-\frac{1}{5})$
$= 1000 \cdot \frac{1}{2} \cdot \frac{4}{5} = 400.$

Proof Sketch:

(1) If $\gcd(m,n)=1$, then $\phi(mn)=\phi(m)\cdot\phi(n)$.

This is because CRT gives a one-to-one mapping between $A \times B$ and C, where

$A = \{$numbers that are coprime with $m\}$
$B = \{$ ...... $n\}$
$C = \{$ ...... $mn\}$

$A \times B = \{(a,b) : a \in A, b \in B\}$
$|A \times B| = |A| \cdot |B|$.

(2) If $n=p^k$ for some prime p, then $\phi(n) = n \cdot (1-\frac{1}{p})$.

Possible values of $\gcd(n,i)$ for $1 \le i \le n$ must be $1, p, p^2, p^3, \dots, p^k$.

$\gcd(n,i)=1$ iff $p \nmid i$. $\Rightarrow \phi(n) = n \cdot (1-\frac{1}{p})$

Example: $3^{84} \bmod 100$.

$\phi(100) = 100 \cdot (1-\frac{1}{2})(1-\frac{1}{5}) = 40 \Rightarrow 3^{40} \equiv 1 \pmod{100}$

$3^{84} \equiv 3^{40} \cdot 3^{40} \cdot 3^4 \equiv 3^4 \equiv 81 \pmod{100}$. $\quad 3^2 \equiv 9$
$3^4 \equiv 9^2 \equiv 81 \pmod{100}$

Example: $2^{2004} \bmod 100$

$\phi(100)=40 \Rightarrow$ ~~$2^{2004} \equiv (2^{40})^{50} \cdot 2^4 \equiv 16 \pmod{100}$.~~ No. Because $\gcd(2,100) \ne 1$.

Euler Theorem: If $\gcd(a,n)=1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.

Q: can we speed up $x^a \bmod n$ when $\gcd(a,n) \ne 1$?

A: Yes. This can be done by using Euler's theorem + CRT. (see HW2 for more detail.)

Encryption Standards

In 1973, NBS (→ NIST) wanted to select a crypto. algorithm as a national standard. (page 113 of textbook).

* secure
* computational efficient

In 1975, NBS published DES.

(see Section 4.2 for a simplified DES-like algorithm).

16 rounds of encryption.

In 1997, NIST put out a call for candidates to replace DES.

Eventually "Rijndael" was chosen as AES.

Symmetric key Encryption: Alice and Bob to agree on a secret key k.

encryption key ≈ decryption key.

Public-key Encryption: (PKE)

Use case: Alice wants to talk to Bob.

They never met before.

Eve is listening from the very beginning.

but Eve cannot decrypt the message.

Bob
- public key (pk) → publishes pk to everyone.
- private key (sk) → keeps sk to himself.

For anyone who wants to send a message m to Bob.

he/she sends $enc(m, pk_{Bob}) \to c$

Only Bob can decrypt c and recover m.

$$dec(c, sk_{Bob}) \to m.$$

A PKE-algorithm consists of:

generate-key(): produces pk and sk

enc():
dec(): $dec(sk, enc(m, pk)) = m$.

Asymmetric: (Ideally) Bob's public key should not reveal any information about his private key.