W3203
Discrete Mathematics

Number Theory

Spring 2015
Instructor: Ilia Vovsha

http://www.cs.columbia.edu/~vovsha/w3203

Outline

- Communication, encryption
- Number system
- Divisibility
- Prime numbers
- Greatest Common Divisor (GCD)
- Euclidean Algorithm
- Modular Arithmetic
- Euler's totient function
- RSA cryptosystem
- Text: Rosen 4
- Text: Lehman 8

Private Communication in Public

- The Problem:
  - Alice (A) wants to tell Bob (B) a military secret. But the enemy (E) is listening to their conversation
  - Can they communicate with each other without revealing the secret to the enemy?
- General approach:
  - Communicate in secret code
  - A & B agree on a procedure to encrypt messages
  - The receiver (B) has a procedure to decrypt the message
  - The enemy (E) should not be able to deduce the decryption procedure

Encryption

- Goal: create a secret code (cipher)
  1. Monographic substitution: permute alphabet, replace each letter by substitute
  2. Shift cipher: represent letters as numbers, shift all letters by some integer, replace with new numbers
- Shift cipher:
  - {A, B, C, … , Y, Z} → {0, 1, 2, … , 24, 25}
  - {0, 1, 2, … , 24, 25} → {3, 4, 5, … , 1, 2}
  - {3, 4, 5, … , 1, 2} → {D, E, F, … , B, C}
  - To decrypt (recover the original), shift back by the same #

Shift Cipher (example)

- Shift cipher:
  - Numbers: {A, B, C, ... , Y, Z} → {0, 1, 2, ... , 24, 25}
  - Shift: {0, 1, 2, ... , 24, 25} → {3, 4, 5, ... , 1, 2}
  - Letters: {3, 4, 5, ... , 1, 2} → {D, E, F, ... , B, C}
- Example: encrypt the message

1. “MEET YOU IN THE PARK”
2. “12 4 4 19 24 14 20 8 13 19 7 4 15 0 17 10”
3. “15 7 7 22 1 17 23 11 16 22 10 7 18 3 20 13”
4. “PHHW BRX LQ WKH SDUN”

Breaking the Code (1)

- Can we discover the message without knowing the encryption method and "key"?
  - Complicated cipher? Difficult to use!
  - Simple cipher? Can't hide patterns!
  - General knowledge can help: relative frequencies of letters
  - Enemy may have access to multiple messages
  - Decryption is computationally feasible

Number System

DEF: The ***natural numbers*** are a mathematical system

$$\{\mathbb{N},\ 0 \in \mathbb{N},\ s : \mathbb{N} \to \mathbb{N}\}$$

with a number ***zero*** 0 and a ***successor*** operation $s : \mathbb{N} \to \mathbb{N}$ such that

(1) $(\nexists n)\,[0 = s(n)]$.
Zero is not the successor of any number.

(2) $(\forall m, n \in \mathbb{N})\,[m \neq n \Rightarrow s(m) \neq s(n)]$.
Different numbers cannot have the same successor.

(3) Given a subset $S \subseteq \mathbb{N}$ with $0 \in S$

$$\text{if } (\forall n \in S)\,[s(n) \in S] \text{ then } S = \mathbb{N}$$

Arithmetic Operations

DEF: The ***predecessor*** of a natural number n is a number m such that $s(m) = n$.
NOTATION: $p(n)$.

DEF: ***Addition*** of natural numbers.

$$n + m = \begin{cases} n & \text{if } m = 0 \\ s(n) + p(m) & \text{otherwise} \end{cases}$$

DEF: ***Ordering*** of natural numbers.

$$n \geq m \text{ means } \begin{cases} m = 0 & \text{or} \\ p(n) \geq p(m) \end{cases}$$

DEF: ***Multiplication*** of natural numbers.

$$n \times m = \begin{cases} 0 & \text{if } m = 0 \\ n + n \times p(m) & \text{otherwise} \end{cases}$$

Division

- Definition: let n and d be integers with $d \neq 0$. If there exists an integer q such that n = dq, then d divides n
  - d is a factor or (proper) divisor of n
  - n is a multiple of d
  - Notation: $d \mid n$ $\quad$ $d \nmid n$
  - Facts: $n \mid 0$ $\quad$ $n \mid n$ $\quad$ $1 \mid n$

Properties of Divisibility

- Properties:

  Let a, b, and c be integers with $a \neq 0$

  1. If a | b and a | c then a | (b+c)
  2. If a | b and b | c then a | c
  3. If a | b and a | c then a | (sb + tc) for all integers s,t

- Proof of part (3):

  a. By definition, $\exists k_1, k_2 \in \mathbb{Z}$: $ak_1 = b$ and $ak_2 = c$
  b. It follows that, $sb + tc = s(ak_1) + t(ak_2) = a(sk_1 + tk_2)$
  c. $sk_1 + tk_2 \in \mathbb{Z} \rightarrow a \mid (sb + tc)$

Division Theorem

- Let n ∈ $\mathbf{Z}$, d ∈ $\mathbf{Z}^+$, then there are unique nonnegative integers *q* and *r* < *d*, such that *n* = *dq* + *r*
  - *d* is called the *divisor*
  - *n* is called the *dividend*
  - *q* is called the *quotient*
  - *r* is called the *remainder*
  - *r* = *n* **mod** *d*

Modular Arithmetic

- Definition: let b and n > 0 be integers. Then b **mod** n is the residue (remainder) of dividing b by n.
- Definition: if a, b, and n > 0 be integers. Then **a** is *congruent to b modulo n* if n divides a – b
- Notation:
  - $a \equiv b \pmod{n}$
  - $a \equiv_{\mathbf{mod\ n}} b$
  - $a \not\equiv b \pmod{n}$
- Congruence modulo n defines a partition of the integers into n sets so that congruent numbers are all in the same set

Shift Cipher (functions)

- Shift cipher: letters shifted by some integer (k)
  - Numbers: {A, B, C, ... , Y, Z} → {0, 1, 2, ... , 24, 25}
  - Shift: {0, 1, 2, ... , 24, 25} → {3, 4, 5, ... , 1, 2}
  - Letters: {3, 4, 5, ... , 1, 2} → {D, E, F, ... , B, C}
- Encryption / Decryption functions (k is the key):
  - $f(p) = (p + k) \bmod 26$
  - $f^{-1}(p) = (p - k) \bmod 26$

Linear Combination

- An integer n is a linear combination of numbers b_0 , ... , b_k iff **$n = c_0b_0 + c_1b_1 + ... + c_kb_k$** for some integers $\{c_0 , ... , c_k\}$
- Application: represent numbers using a linear combination to improve efficiency of algorithms
- Common representation: decimal, or *base* 10
- We can represent numbers using any base b, where b is a positive integer greater than 1
- The bases b = 2 (binary), b = 8 (octal) , and b= 16 (hexadecimal) are important for computing and communications
- Example: $965 = 9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$

Base b Representations

- ***Theorem***: let *b* be a positive integer greater than 1. Then if *n* is a positive integer, it can be expressed uniquely in the form:

$$\mathbf{n = a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 b + a_0}$$

where *k* is a nonnegative integer, $a_0, a_1, \ldots, a_k$ are nonnegative integers less than *b*, and $a_k \neq 0$.

The a_j, $j = 0, \ldots, k$ are called the base-*b* digits of the representation

- Example: $965 = 9 \cdot 10^2 + 6 \cdot 10^1 + 5 \cdot 10^0$

Base b Expansions (examples)

- What is the decimal expansion given the binary expansion?
  - $(1\ 0101\ 1111)_2 = 1\cdot 2^8 + 0\cdot 2^7 + 1\cdot 2^6 + 0\cdot 2^5 + 1\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 351$
- What is the "decimal given binary" expansion?
  - $(11011)_2 = 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = 27$
- What is the "decimal given octal" expansion?
  - $(7016)_8 = 7\cdot 8^3 + 0\cdot 8^2 + 1\cdot 8^1 + 6\cdot 8^0 = 3598$
- What is the "octal given decimal" expansion?
  - $(12345)_{10} = (30071)_8$

Turing's Code (not really)

- Approach:
  - Convert message from letters to positive integers (e.g. standard ASCII code)
  - Combine separate numbers into one large integer **M**
  - Pad the result (**M**) with more digits to make a prime number (**p**)
  - Multiply **p** by a large prime number **k** (a secret key agreed to beforehand but unknown to the enemy)
  - Send message $\mathbf{M^* = p \times k}$
  - Receiver decrypts message by computing $\mathbf{p = M^* / k}$, and deducing the words from the sequence of letters (**M**)

Turing's Code (example)

- Example:
  1. *Translate*: {A, B, C, ... , Y, Z} → {01, 02, 03, ... , 25, 26}
  2. *Message*: "victory" → {22 09 03 20 15 18 25}
  3. *Pad to prime*:
  4. {22 09 03 20 15 18 25} → 2209032015182513
  5. *Secret key*: **k** = 22801763489
  6. **M^* = p x k**

     = 2209032015182513 x 22801763489

     = 50369825549820718594667857

Prime Numbers

- Definition: a positive integer $p > 1$ is called *prime* if the only positive factors of p are 1 and p. A positive integer > 1 which is not prime is called *composite*
- Prime questions:
  - How many primes are there?
  - Can we efficiently determine whether a number is prime?
  - What is the distribution of prime numbers?
  - How can we generate large primes?
  - Can we efficiently factor composite numbers into their prime factorizations?

How Many Primes?

- Theorem: there are infinitely many primes.
- Proof:
  - Suppose there are finitely many primes: $\{p_1, \ldots, p_k\}$
  - Let $q = p_1 p_2 \cdots p_k + 1$
  - Either q is prime or it is composite (product of primes)
  - By assumption it is composite
  - But none of the primes p_j divides q since if $p_j \mid q$, then p_j divides $q - p_1 p_2 \cdots p_k = 1$
  - Hence, there is a prime not on the list $\{p_1, \ldots, p_k\}$ which is a prime factor of q
  - Contradiction!

Prime Factorization

- **Fundamental Theorem of Arithmetic**: every positive integer is a product of a unique *weakly decreasing sequence* of primes (*prime factorization*).
- Proof idea:
  - Assume the factorization is not unique
  - Define two sequences (for both, the product equals n)
  - Compare the largest prime factor in each sequence
  - w.l.o.g, you can divide n by the larger of these (call it 'f')
  - Derive contradiction with the fact that 'f' is the largest prime factor

Primality Testing

- Given an integer n, is it prime?
- Naive Algorithm: for each d ∈ $[2,n-1]$, if d | n, then stop and return "FALSE"
- Less Naive Algorithm: for each d ∈ $[2,\sqrt{n}]$, if d | n, then stop and return "FALSE"
- *Probabilistic test*: gives the right answer when applied to any prime number, but has some (very tiny) probability of giving a wrong answer on a nonprime number

Distribution of Primes

- Primes show up erratically, but we can give an *asymptotic* estimate for the number of primes not exceeding some integer n
- ***Prime Number Theorem***: the ratio of the number of primes π(n) not exceeding n and n/ln n approaches 1 as n grows without bound.

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

- As a rule of thumb, about 1 integer out of every ln n in the vicinity of n is a prime (odds of random selection)

Breaking the Code (2)

- Can we discover the message without knowing the "key"?
  - Recovering the original message requires factoring a very large number into its prime factors
  - Conjecture: there is no computationally efficient procedure for prime factorization
  - But enemy may have access to multiple messages!
  - Message 1: $\mathbf{M^*_1 = p_1 \times k}$ Message 2: $\mathbf{M^*_2 = p_2 \times k}$
  - The key (k) divides both $\mathbf{M^*_1}$, $\mathbf{M^*_2}$
  - Compute the greatest common divisor of $\mathbf{M^*_1}$, $\mathbf{M^*_2}$

Greatest Common Divisor

- Definition: let *a* and *b* be integers, not both zero. The largest integer *d* such that *d* | *a* and also *d* | *b* is called the *greatest common divisor* of *a* and *b,* denoted by *gcd(a,b)*
- Examples:
  - $\gcd(24,36) = 12$
  - $\gcd(17,22) = 1$
  - $\gcd(n, 0) = n$
- Definition: the integers *a* and *b* are *relatively prime* if their gcd is 1, $a \perp b$

Computing the GCD

Algo 4.3.4: Primepower GCD Algorithm

Input: integers $m \leq n$ not both zero
Output: $\gcd(m, n)$

(1) Factor $m = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$ into prime powers.

(2) Factor $n = p_1^{b_1} p_2^{b_2} \cdots p_r^{b_r}$ into prime powers.

(3) $g := p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \cdots p_r^{\min(a_r,b_r)}$

Return (g)

Least Common Multiple

- Definition: let *a* and *b* be positive integers. The least common multiple of *a* and *b* is the smallest positive integer that is divisible by both *a* and *b*, denoted by *lcm(a,b)*

$$\operatorname{lcm}(a,b) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \cdots p_n^{\max(a_n,b_n)}$$

- Example: $\operatorname{lcm}(2^3 3^5 7^2, 2^4 3^3) = 2^{\max(3,4)}\, 3^{\max(5,3)}\, 7^{\max(2,0)}$

$= 2^4\, 3^5\, 7^2$

- Fact: $ab = \gcd(a,b) \cdot \operatorname{lcm}(a,b)$

Euclid's Observation

- Observation: let ***a = bq + r***, where a, b ≠ 0, q, and r are integers. Then, ***gcd(a,b) = gcd(b,r)***
- Proof:
  a. By definition, **a** is a linear combination of ***b*** and ***r***. Likewise, **r** is a linear combination combination, ***a - qb***, of ***a*** and ***b***.
  b. It follows that any divisor of ***b*** and ***r*** is a divisor of **a**. Any divisor of **a** and ***b*** is a divisor of ***r***.
  c. It follows that **a** and ***b*** have the same common divisors as ***b*** and ***r***.
  d. Hence they have the same *greatest* common divisor ***gcd(a,b) = gcd(b,r) = gcd(b, a mod b)***

Euclidean Algorithm

Algo 4.3.5: Euclidean Algorithm

Input: positive integers $m \geq 0, n > 0$
Output: $\gcd(n, m)$

If $m = 0$ **then return**(n)
else return $\gcd(m, n \bmod m)$

$$\begin{aligned}
\gcd(210, 111) &= \gcd(111, 210 \bmod 111) = \\
\gcd(111, 99) &= \gcd(99, 111 \bmod 99) = \\
\gcd(99, 12) &= \gcd(12, 99 \bmod 12) = \\
\gcd(12, 3) &= \gcd(3, 12 \bmod 3) = \\
\gcd(3, 0) &= 3
\end{aligned}$$

Euclidean Algorithm (example)

Example 4.3.6: Euclidean Algorithm

$$\begin{aligned}
\gcd(42,26) &= \gcd(26,42 \bmod 26) = \\
\gcd(26,16) &= \gcd(16,26 \bmod 16) = \\
\gcd(16,10) &= \gcd(10,16 \bmod 10) = \\
\gcd(10,6) &= \gcd(6,10 \bmod 6) = \\
\gcd(6,4) &= \gcd(4,6 \bmod 4) = \\
\gcd(4,2) &= \gcd(2,4 \bmod 2) = \\
\gcd(2,0) &= 2
\end{aligned}$$

Extended Euclidean Algorithm

- The greatest common divisor of a and b is a linear combination of a and b. That is, for some integers s and t (*Bézout coefficients*): ***gcd(a,b) = sa + tb***
- How do you determine s and t?

$a = r_0 \quad b = r_1$

$r_0 = r_1 q_1 + r_2 \qquad 0 \le r_2 < r_1,$

$r_1 = r_2 q_2 + r_3 \qquad 0 \le r_3 < r_2,$

.

.

.

$r_{n-2} = r_{n-1} q_{n-1} + r_n \qquad 0 \le r_n < r_{n-1},$

$r_{n-1} = r_n q_n .$

$\gcd(a,b) = r_n$

$r_n = r_{n-2} - r_{n-1} q_{n-1}$

$r_{n-1} = r_{n-3} - r_{n-2} q_{n-2}$

.

.

.

$r_3 = r_1 - r_2 q_2 = b - (a - b q_1) q_2$

$r_2 = r_0 - r_1 q_1 = a - b q_1$

Extended Euclidean Algorithm (example)

- ***gcd(a,b) = sa + tb***
- Example: gcd(259,70)

259 = 70 x 3 + 49 **49 = 259 - 70 x 3**

70 = 49 x 1 + 21 21 = 70 - 49 x 1

= 70 – (259 - 70 x 3) x 1

= -(259 x 1) + (70 x 4)

49 = 21 x 2 + 7 **7 = 49 - 21 x 2**

= (259 - 70 x 3) – [-(259 x 1) + (70 x 4)] x 2

= [3 x 259] - [11 x 70]

21 = 7 x 3

Turing's Code (better idea)

- Approach:
  - Convert message to into one large integer **M** and pad to make a prime number **p**
  - Choose a large prime number **n** > p (n can be made public)
  - Multiply **p** by a large prime number **k** < n (k is a secret key)
  - Send message $\mathbf{M^* = (p \times k) \bmod n}$
  - ~~Receiver decrypts message by computing **$p = M^* / k$**~~
  - Decryption is a problem! Must compute "inverse mod n"

Turing's Code (example)

- Example 1:
  1. *Message:* $p = 5$
  2. *Large prime:* $n = 17$ *Secret key*: $k = 13$
  3. **$M^* = (p \times k) \bmod n$**
     = 65 mod 17
     = 14
- Example 2:
  1. *Message:* $p = 7$
  2. *Large prime:* $n = 17$ *Secret key*: $k = 13$
  3. **$M^* = (p \times k) \bmod n$**
     = 91 mod 17
     = 6

Multiplicative Inverse

- Definition: the *multiplicative inverse* of a number x is another number x^{-1} such that: $x^{-1}\, x = 1$
  - Except 0, every rational number n / m has an inverse, namely, m/n.
  - Over the integers, only 1 and -1 have inverses
- What about modular arithmetic ("ring $\mathbf{Z_n}$")?
  - $(2 \cdot 8) \bmod 15 \;=\; 2 \cdot_n 8 \;=\; 1$
  - $(? \cdot 3) \bmod 15 \;=\; ? \cdot_n 3 \;=\; 1$
  - Some numbers have inverses modulo 15 and others don't

Modular Arithmetic Rules

1. $a \equiv \text{rem}(a, n) \pmod{n}$ $\qquad$ $a \equiv_{\text{mod } n} \text{rem}(a, n)$

2. If $\mathbf{a} \equiv_{\text{mod } n} \mathbf{b}$ and $\mathbf{c} \equiv_{\text{mod } n} \mathbf{d}$, then
   I. $a + c \equiv_{\text{mod } n} b + d$
   II. $ac \equiv_{\text{mod } n} bd$

Modular Arithmetic Rules (2)

1. $a \equiv \text{rem}(a, n) \pmod n$ $\qquad a \equiv_{\text{mod } n} \text{rem}(a, n)$
2. If $a \equiv_{\text{mod } n} b$ and $c \equiv_{\text{mod } n} d$, then
   - I. $a + c \equiv_{\text{mod } n} b + d$
   - II. $ac \equiv_{\text{mod } n} bd$
3. Define operations in $\mathbf{Z_n}$: $\cdot_n$ $+_n$
   - $a +_n b ::= \text{rem}(a + b, n)$ $\qquad a \cdot_n b ::= \text{rem}(a \cdot b, n)$
   1. $(a + b) \bmod n = [\,(a \bmod n) + (b \bmod n)\,] \bmod n$

      $\text{rem}(a + b, n) = \text{rem}(a, n) +_n \text{rem}(b,n)$
   2. $(ab) \bmod n = [\,(a \bmod n) \cdot (b \bmod n)\,] \bmod n$

      $\text{rem}(ab, n) = \text{rem}(a, n) \cdot_n \text{rem}(b,n)$

Modular Arithmetic (example)

- Find: $\mathrm{rem}\left((44427^{3456789} + 15555858^{5555})403^{6666666}, 36\right).$
- Use rules:
  1. **rem(a + b, n) = rem(a, n) $+_n$ rem(b,n)**
  2. **rem(ab, n) = rem(a, n) $\cdot_n$ rem(b,n)**
- Simplify:

  rem(44427,36) = 3, rem(15555858, 36) = 6, rem(403,36) = 7

  $$(3^{3456789} + 6^{5555})7^{6666666}$$
  $$(3^3 + 6^2 \cdot 6^{5553})(7^6)^{1111111}$$
  $$(3^3 + 0 \cdot 6^{5553})1^{1111111}$$
  $$= 27.$$

Inverse in $\mathbf{Z_n}$

- Definition: the *multiplicative inverse* of a number x is another number x^{-1} such that: $x^{-1}\, x = 1$
- What about modular arithmetic ("ring $\mathbf{Z_n}$")?
  - $x \cdot a \equiv 1 \bmod n$

    $\rightarrow \quad xa - qn = 1$

    $\rightarrow \quad \gcd(a, n) = 1$
  - Conclusion: for a number ('a') to have an inverse in $\mathbf{Z_n}$, '**a**' must be relatively prime to n

Turing's Code (Decryption)

- Approach:
  - Convert message to into one large integer **M** and pad to make a prime number **p**
  - Choose a large prime number **n** > p (n can be made public)
  - Multiply **p** by a large prime number **k** < n (k is a secret key)
  - Send message $\mathbf{M^* = (pk) \bmod n}$
  - Receiver decrypts message by computing the $\mathbf{Z_n}$**-** inverse **j** of the key **k** using the extended Euclidean algorithm:

$$M^* \cdot_n j = (p \cdot_n k) \cdot_n j = p \cdot_n (k \cdot_n j) = p \cdot_n 1 = p$$

Breaking the Code (3)

- Can we discover the message without knowing the key?
  - Enemy may have access to multiple messages. No problem, we are working in $\mathbf{Z_n}$
  - Suppose the enemy knows both the message (plaintext), **M**, and its encrypted form, $\mathbf{M^*}$
  - Enemy carries out a *known-plaintext attack!*
  - $\mathbf{M^* = p \cdot_n k}$ $\quad n > p \quad n > k$
  - Using the extended Euclidean algorithm, enemy computes the $\mathbf{Z_n}$- inverse **j** of **p** and obtain the secret key:

$$j \cdot_n M^* = j \cdot_n (p \cdot_n k) = (j \cdot_n p) \cdot_n k = 1 \cdot_n k = k$$

Public Key Cryptography

- Approach:
  - Convert message into one large integer **M**
  - The receiver privately creates a pair of functions: **E** to encrypt the message, and **D** to decrypt the message, such that **D[E(M)] = M**
  - Receiver publicly reveals the function **E**
  - Message is sent: $\mathbf{M}^* = \mathbf{E(M)}$
  - Enemy can see $\mathbf{M}^*$ and knows **E** but can't determine **D**

RSA (idea)

- A public key cryptosystem was introduced in 1976 by three researchers at MIT: Rivest, Shamir, Adelman
- Idea:
  - Convert message into one large integer **M**
  - Receiver finds two large primes **p, q** (using probabilistic primality tests) and calculates their product ***n = pq*** ***(n > M)***
  - Receiver finds two integers **e, d** and creates a pair of functions:

    $E(M) = M^e \bmod n$ to encrypt the message

    $D(M^*) = (M^*)^d \bmod n$ to decrypt the message
  - Receiver publicly reveals **E (n & e)**
  - Message is sent: $M^* = E(M)$
  - Enemy can see M^* and knows **E** but can't determine **d**

RSA (setup)

- Idea:
  - Convert message into one large integer **M**
  - Receiver finds two large primes **p, q,** their product, ***n = pq***
  - Receiver finds two integers **e, d** and creates a pair of functions:

    $E(M) = M^e \bmod n$ to encrypt the message

    $D(M^*) = (M^*)^d \bmod n$ to decrypt the message
  - Receiver publicly reveals **E (n & e)**
  - Message is sent: **$M^* = E(M)$**
  - Enemy can see **M^*** and knows **E** but can't determine **d**
  - System only works if: **$D(M^*) = D[\,E(M)\,] = M$**
  - **$D[\,E(M)\,] = D(\,M^e\,) = (\,M^e\,)^d = M^{ed} = M$** working in in Z_n

Euler's Totient Function

- Definition: let *φ(n)* be defined as the number of integers in **[0,n)** that are relatively prime to n > 0.
- Examples:
  - $\phi(12) = 4$ $\{1,5,7,11\}$
  - $\phi(7) = 6$ $\{1,2,3,4,5,6\}$
  - $\phi(11) = 10$
- Rules:
  1. If ***p*** is prime, $\phi(p) = p - 1$
  2. If ***p*** ≠ ***q*** are both primes, $\phi(pq) = (p - 1)(q - 1)$
  3. If ***a*** and ***b*** are relatively prime, $\phi(ab) = \phi(a)\phi(b)$

Euler's Theorem

- Definition: let *φ(n)* be defined as the number of integers in **[0,n)** that are relatively prime to n > 0.
- *Euler's Theorem*: if n and k are relatively prime, then:

$$k^{\varphi(n)} \equiv 1 \pmod{n}$$

- Recall: if ***p*** is prime, $\phi(p) = p - 1$
- *Fermat's Little Theorem*: if ***p*** is prime, and ***k*** is not a multiple of p, then:

$$k^{p-1} \equiv 1 \pmod{p}$$

RSA (derivation)

- Recall:
  - $n = pq$
  - System only works if: $D(M^*) = D[E(M)] = M$
  - $D[E(M)] = D(M^e) = (M^e)^d = \boldsymbol{M^{ed} = M}$ ***working in in*** $\boldsymbol{Z_n}$
- Derivation:

1.	$n \perp M,\ M^{\varphi(n)} \equiv 1 \pmod n$	Euler's Theorem, $\gcd(M, n) = 1$
2.	$M^{c\varphi(n)} \equiv 1 \pmod n$	Modular Arithmetic
3.	$M^{c\varphi(n)+1} \equiv M \pmod n$	Modular Arithmetic
4.	$\phi(n) = \phi(pq) = (p-1)(q-1)$	Rule
5.	$e \cdot d = c \cdot \phi(n)+1 \rightarrow ed \equiv 1 \pmod{\phi(n)}$	
6.	$\gcd(e, \phi(n)) = 1 \rightarrow \gcd(e, (p-1)(q-1)) = 1$	
7.	**d** is the $\mathbf{Z}_{\phi(n)}$- inverse of **e**	

RSA Cryptosystem

1. The Receiver prepares the system as follows:
   a. Generates two large distinct primes **p, q,** keeps them private
   b. Calculates the product, ***n = pq,*** makes it public
   c. Selects and integer **e ∈ [0,n)**, such that gcd(e, (p – 1) (q – 1)) = 1, makes it public
   d. Calculates an integer **d ∈ [0,n)** which is the $\mathbf{Z_{(p-1)(q-1)}}$ **-** inverse of ***e,*** using the extended Euclidean algorithm, keeps ***d*** private
2. Sender prepares and publicly transmits message:
   a. Converts message into one large integer **M ∈ [0,n)** such that gcd(M, n) = 1
   b. Encrypts message using public key, $\mathbf{M^* = E(M) = M^e \bmod n}$
3. Receiver privately decrypts message:
   a. Decrypts message using private key, $\mathbf{M = D(M^*) = (M^*)^d \bmod n}$

RSA Cryptosystem (example)

1. The Receiver prepares the system as follows:
   a. Generate: **p = 1231, q = 337**
   b. Calculate: ***n = pq = 414847,*** ***(p-1)(q-1) = 413280***
   c. Select integer **e ∈ [0,n)**: **e = 211243**
   d. Calculate integer **d ∈ [0,n)**: **$d = e^{-1} = 166147$**
2. Sender prepares and publicly transmits message:
   a. Converts message: **M = 224455**
   b. Encrypts message: **$M^* = E(M) = M^e \bmod n = 376682$**
3. Receiver privately decrypts message:
   a. Decrypts message: **$M = D(M^*) = (M^*)^d \bmod n = 224455$**