Modulo

Remainder when dividing two positive integers (mod as an operator):

- \( n = m \cdot \lfloor n/m \rfloor + (n \mod m) \)

Extended to negative numbers (\( y \neq 0 \))

- \( x \mod y = x - y \cdot \lfloor x/y \rfloor \)

Equivalence classes (mod as an equivalence relation)

- \( a \equiv b \mod m \) iff \( a - b \) is a multiple of \( m \)

Residue classes mod \( m \)

- \( m \) of them
  - All \( a \) such that \( a \equiv 0 \mod m \)
  - All \( a \) such that \( a \equiv 1 \mod m \)
  - \( \ldots \)
  - All \( a \) such that \( a \equiv m-1 \mod m \)

Modulo Arithmetic

If:

- \( x = x' \mod m \)
- \( y = y' \mod m \)

Then

- \( x + y \equiv x' + y' \mod m \)
- \( x - y \equiv x' - y' \mod m \)
- \( xy \equiv x'y' \mod m \)

Applications

- Casting out 9’s

\[
\begin{array}{c|c|c}
532 & + & 656 \\
327 & 55 & 17985 \\
95 & 1273 & \\
\end{array}
\]

gcd, lcm, \( \Phi \)

Greatest Common Divisor (gcd)

- \( \gcd(m, n) \) is the largest integer \( k \) that divides integers \( m \) and \( n \)
  - \( k \mid m \) and \( k \mid n \)
- \( \gcd(m, n) \) is a linear combination (with integer coefficients) of \( m \) and \( n \)
  - \( \exists i, j \in \mathbb{Z}: \gcd(m, n) = im + jn \)
- To calculate \( \gcd(m, n) \)
  - Compute prime factorization of \( m \) and \( n \)
  - \( \gcd(m, n) = \text{common prime factors (and powers) of } m \text{ and } n \)
- Euclid’s algorithm
  - int \( \gcd(m, n) \)
  - if \( (n == 0) \) return \( m \)
  - else return \( \gcd(n, m \mod n) \)

Least Common Multiple (lcm)

- \( \text{lcm}(m, n) \) is the smallest integer \( k \) such that integers \( m \) and \( n \) divide \( k \)
  - \( m \mid k \) and \( n \mid k \)
- To calculate \( \text{lcm}(m, n) \)
  - Compute prime factorizations of \( m \) and \( n \)
  - \( \text{lcm}(m, n) = \text{union of prime factors (and powers) of } m \text{ and } n \)

\( \Phi \) (Euler function)

- \( \Phi(n) \) = the number of positive integers \( k \leq n \) such that \( k \perp n \) \( (\gcd(k, n) = 1) \)
Properties of $\Phi$

If $n$ is prime
- $\Phi(n) = n - 1$

If $n$ is product of two distinct primes, $p$ and $q$
- $\Phi(n) = (p-1)(q-1)$
- Number not relatively prime = $q + p - 1$.
- $pq - (q + p - 1) = (p-1)(q-1)$

Call the $\Phi(n)$ numbers relatively prime to $n$ units
- Given a unit of $n$, $a: a^{\Phi(n)} \equiv 1 \pmod{n}$

Thus, given two distinct primes, $p$ and $q$
- $m \perp pq \rightarrow m^{(p-1)(q-1)} \equiv 1 \pmod{pq}$

Cryptography

Definitions:
- Plaintext: message being encoded
- Ciphertext: encoded plaintext
- Key: parameter to crypto algorithms
  - $C = E(P, K_E)$
  - $P = D(C, K_D)$

Simple cipher (Caesar cipher):
- $K_E = K_D$ = permutation from letters to letters
  - $A \rightarrow X$
  - $B \rightarrow L$
  - $Z \rightarrow R$
- $E = \text{apply } K_E$ to each character in plaintext
- $D = \text{apply } K_E^{-1}$ to each character in ciphertext
- Weakness: letter frequencies

Unbreakable Code

One-time pad
- $K_E = K_D$ = long stream of random bytes
  - Really random, not pseudo-random
- $E(P, K_E)$
  - for $i = 1$ to length($P$)
    - $C[i] = P[i] \text{ XOR } K_E[i]$
- $D = E$

Alice sends message to Bob
- Uses secret one-time pad
  - Encrypts $P$
  - Destroys $P$ and one-time pad

Bob decrypts
- Using one-time pad

Alternative encryption (by-hand)
- Modular arithmetic
- Weakness
  - Key must be as long as plaintext
  - Key must be used only once
  - Key must be truly random

Reusing Key

Problem
- If plaintext and ciphertext are both known, key can be reverse-engineered
  - $K_E[i] = P[i] \text{ XOR } C[i]$
- How to know plaintext if it is encrypted?
  - Cause specific plaintext to be sent
    - British would mine specific areas in WWII so that the Germans would send
      message including “minen” and location.
Trapdoor Functions

Trapdoor function: Given the output of a function, the input is hard to compute

Discrete logarithms
- Consider computing a power of a number modulo a prime
  - 3^t mod 7
  - Easy to compute
- Consider the reverse problem, given 3^t mod 7, what is t?
  - Believed to be difficult to compute for large numbers
  - t is the discrete logarithm

Can be used to re-use keys
- Instead of encrypting with K, pick random b and encrypt with \( L = b^K \mod p \) (known prime p)
- Send C as well as b
- If attacker knows plaintext, can figure out L, but that doesn’t help figure out K

Cryptography without a Shared Key

Symmetric Key Exchange (Diffie-Hellman)
- Alice and Bob don’t have a shared key, but want to exchange information such that they know a shared key but attackers don’t.
- Alice and Bob agree on prime p and base z (1 < x < p-1)
  - Standard numbers, perhaps
- Alice picks a random number a (1 < a < p-1), computes \( A = z^a \mod p \)
- Bob picks a random number b (1 < b < p-1), computes \( B = z^b \mod p \)
- Alice and Bob exchange A and B
  - Everyone knows A, B, p, and z
- Alice computes \( K = B^a \mod p \)
- Bob computes \( K = A^b \mod p \)

- \( B^a \equiv (z^b)^a \equiv z^{ab} \equiv (z^a)^b \equiv A \mod p \)

Weakness/strength:
- If n can be factored, private key can be easily recreated

Public Key Cryptography (RSA)
- Alice has a public key, KE, used for encrypting messages to her
- She has a secret private key, KD, used for decryption
- Creating pair of keys:
  - Choose two large random primes p, and q
  - Compute n = pq
  - Compute \( \phi(n) = (p-1)(q-1) \)
  - Choose an integer 1 < e < \( \phi(n) \) such that e \( \perp \phi(n) \)
  - Compute d such that de \( \equiv 1 \mod \phi(n) \)
- \( K_E = (n, e), K_D = (n, d) \)
- Choose P with 1 ≤ P < n, and P \( \perp n \)
- Encrypt: \( C = P^e \mod n \)
- Decrypt: \( P = C^d \mod n \)
  - \( C^e \equiv (P^d)^e \equiv P^d \mod n \)
  - Since de \( \equiv 1 \mod \phi(n) \), de \( \equiv 1 + i \phi(n) \)
  - So, \( P^d \equiv P^{1+i\phi(n)} \equiv P^{1+ir}\Phi(n) \equiv P(\Phi^{(r)}(n) \mod n) \)
  - But, \( \Phi^{(r)}(n) \equiv 1 \mod n \), so \( C^e \equiv P \mod n \)
- Weakness/strength:
  - If n can be factored, private key can be easily recreated