- Symmetric Cryptosystem vs Asymmetric Cryptosystem
    - Problems
        - 1 transmission
        - 2 security
        - 3 n choose 2 pair of keys
    - How Asymmetric Cryptosystem solves these problems:
        - 1 transmission
        - 2 security: hard number theory
        - 3 everyone has just one set of public/private keys
- Number Theory:
    - definition(d|a, composite vs prime, common divisor, gcd)
        - propositions(1: a|b, b|c, so a|c, 2: a|b, b!=0, a<=b, 3: d|a, d|b, d|(ra+sb),4: 1 divides a,b, 5: !a=0 & b=0, there are finitely many common divisors)
    - Division lemma
        - Prove Existence(r=a-qn, pick the best q, r must be 0<=r<n)
        - Prove uniqueness(subtract, n<r’-r, contradiction)
    - Thm: gcd(a,b) = min{xa+yb: x,y in Z, xa+yb >0}
        - Proof
            - d is a common divisor of a,b
                - Suppose d is the smallest x’a+y’b, and 0<r<d, contradiction! So r = 0, d|a
            - d is the greatest
                - Suppose z|a, z|b, z|x’a+y’b -> z<=b
        - Corr
            - xa+yb=1 ⇔ gcd(a,b)=1 ⇔ a,b rela prime
    - Prop: gcd(a,b) = gcd(b,r)
        - Proof
            - x|a & x|b iff x|b & x|r
    - Euclid’s Algorithm
        - Algo: while(a_i != 0) {compute a_i-1 = q*a_i + a_i+1; i++} output a_i-1
            - a_i strictly decreases, so terminates eventually
        - r < ½ a (if a <= n)
            - Proof(b<=1/2a, b>1/2a: a = 1*b + (a-b), a-b<b)
        - Number of steps in Euclid Alg is <= 2log_2 a
            - a_6 < ½ a_4 < ¼ a_2 < ⅛ a_0
            - a_2j < (½)^j a
            - #steps <= 2log_2 a(a_2log2 a<(½)^log2a = 1) <= integrity

  - Extended Euclid Algorithm
    - x_j+1=q_jx_j+x_j-1 & y_j+1=q_jy_j+y_j-1
    - a_j = (-1)^j x_j*a + (-1)^j+1 y_j*b
    - Prove a_j = (-1)^j x_j*a + (-1)^{j+1}y_j*b
      - Induction, express a_j+1 as -q_j*a_j+a_j-1
  - FTA
    - Prove Existence
      - induction(prime case, not prime case)
    - Thm: if p prime, p|ab -> p|a or p|b
      - Prove: (suppose p!|b, then 1=xp+yb, multiply by a, p|a)
    - If p|q1q2q3...qm, then p=qi
      - Prove by induction and previous Thm
    - Prove Uniqueness
      - Prove by contradiction and previous Thm
    - p|ab -> p|a and p|b
      - Proof
    - gcd(a,b) = Multiplication(p_i^Min{e_i,d_i})
  - Fermat's Little Theorem
    - p prime, a in Z_p*, a^(p-1) = 1 mod p
  - Euler's Theorem
    - n >1, a in Z_n*, a^(phi(n)) = 1 mod n
    - Cor: a^(-1) = a(phi(n)-1) mod n
    - proof: order(a)|phi(n)
  - Carmichael Numbers
    - when do we use them?
- Abstract Algebra:
  - a congruent to b mod n
    - Definition (n|a-b)
      - mod: a = q*n+r; cong: n|a-b
    - Characteristics
      - a cong a mod n
      - a cong b mod n => b cong a mod n
      - a cong b mod n => b cong c mod n => a cong c mod n
    - a cong b mod n & c cong d mod n
      => a + c cong b + d mod n & a*c cong b*d mod n
  - Z_n groups
    - a + nZ(congruence classes),   Z/nZ
    - (a + nZ) + (b + nZ) = (a + b) + nZ, (a + nZ)*(b + nZ) = (a*b) + nZ
      - because a+nZ = a'+nZ ⇔ a cong a' mod n

        - Isomorphism between a and a+nZ
    - Abelian Groups
        - Closed, associ(a+(b+c)=(a+b)+c), identity(a*epsilon = epsilon*a = a), inverse(a*b = b*a = epsilon), group, abelian(AB=BA, must satisfy group property)
            - Rubik's cube example
        - Suppose G = (V,0) is a group
            - Unique identity
            - Unique inverse
        - Definition of unit
            - a is a unit in n iff gcd(a,n) = 1
        - Definition of Z_n*
            - Z_n*,* is a group
        - Definition of phi(n)
    - Chinese Remainder Thm(n1...nk rela prime, x solves x cong b_i mod n_i)
        - x cong Summation(b_i*N_i*N_i_inverse)
        - Proof: if x = x' mod n1, n2, n3...nk, then x = x' mod n1*n2..=n
            - n1|x - x', n2|x - x'... since n1, n2… rela prime, n|x-x'
    - Definition of subgroup
        - alpha^0 = epsilon
        - <alpha>: subgroup of G generated by alpha(alpha*alpha...)
        - order(alpha) = min{i in Z: alpha^i = epsilon}
        - Prop: order(alpha) < infinity, <alpha> isomorphic to (Z_order(alpha),+), alpha^(order(alpha)-1) = alpha^(-1)
        - Prove prop(2. alpha^(s+t)=alpha^(qm+r))
    - Lagrange Thm: |H| | |G| (proven by propositions below)
    - Left coset
        - Each left coset has cardinality |H|
        - Left coset partition G
        - Proof
            - g^(-1)gh1=g^(-1)gh2 -> h1=h2
            - g1h1=g2h2 ---> g1H belongs to g2H
    - Summation(phi(d) = n) (d|n)
        - Proof 1(paired off),2(1<=x<=n/b bijection 1<=x<=n),3(U=gcd(a,n)=d)
        - Cor:phi(pq)=(p-1)(q-1)
- Computational Complexity
    - 2^x is O(e^x) but e^x is not in O(2^x)
    - length(n) = Theta(log n)
        - d^k <= n <=d^(k+1)

        - k <= logd n <= k+1
    - Running time = Theta(size of input) -> efficient!
    - Example 1: Euclid's Alg -> efficient
        - Theta(x)
    - Example 2: Bruce force primality testing -> exponential
        - Theta(2^x)
        - Theta(sqrt(2)^x) ---modified
- Fast Exponentiation
    - 1. write b as b_i*2^i (remainder is for i=0...k, divide until quotient is 0)
    - 2. 5^(2^a)*5^(2^b)*5^(2^c)...
    - k operations -> efficient
- Ciphers
    - RSA
        - To do list(find large primes, find units in phi(n), security)
        - public: (n,e) private: (p,q,d) (m^e)^d mod n = m mod n
        - What is the prob that n is not invertible
            - (n-invertible)/n = (pq - (p-1)(q-1))/pq = (p+q-1)/pq
        - Digital signatures
            - Authentication, nonrepudiation, efficiency
            - Why cannot forge signature
                - s = m^dA m=s^eA, as hard as RSA!
    - Rabin
        - m^2 cong c mod p, no other square roots besides m and -m
            - Prove by contradiction, suppose a^2 = c mod p, p|a^2 - m^2, p|(a-m)(a+m) -> a=m mod p or a=-m mod p
        - p cong 3 mod 4, square roots are c^((p+1)/4)
            - Proof: (c^((p+1)/4))^2 = c mod p
        - p,q distinct primes, how to find four sq roots of pq
            - m1 = c^((p+1)/4) mod p, m1 = c^((q+1)/4) mod q
            - m2 = c^((p+1)/4) mod p, m1 = -c^((q+1)/4) mod q
            - m3 = -c^((p+1)/4) mod p, m1 = c^((q+1)/4) mod q
            - m4 = -c^((p+1)/4) mod p, m1 = -c^((q+1)/4) mod q
            - Proof: m^2 = c mod pq => pq|m^2-c => p|m^2-c & q|m^2-c => m^2 = c mod p, m^2 = c mod q =>
              m = +- c^((p+1)/4) mod p, m = +- c^((q+1)/4) mod q
        - Efficient algo for computing 4 distinct sq roots provides and efficient factorization of pq

            - $m1^2=c \bmod pq$, $m2^2=c \bmod pq \Rightarrow m1^2=m2^2 \bmod pq \Rightarrow pq|(m1+m2)(m1-m2)$ [p,q must one in (m1+m2), one in (m1-m2)] $\Rightarrow \gcd(pq,(m1-m2)) = p$ or q
            - Proof
    - Elgamal
        - Primitive root definition($<r>=Z_p^*$, r in Z_p^*)
        - All primes p have a primitive root
        - Discrete logarithm
            - No efficient algorithm for computing dlogr
        - Diffie Hellman Key Exchange
            - $A = r^a \bmod p$, $B = r^b \bmod p$
            - $k = A^b$ or B^a but Eve cannot know k,a,b
            - Problem: find k efficiently from p,r,A,B
        - Elgamal Cryptosystem
            - $c = km \bmod p$, $m = k^{(-1)}c \bmod p$
            - Bob inverts k
                - Use Euclid's Algo
                - $k^{(-1)}=A^{(p-1-b)} \bmod p$
- Factorization
    - Running time
        - $2^r \le p1p2...pr=n \Rightarrow r<\log_2 n$
        - r is in x, so running time is xP(x) if factoring is in polynomial time
    - Factoring
        - Trial division
            - Method: divide 1...sqrt(n)
            - Analysis: not efficient
                - $\sqrt{e}^x$
                - Even only check primes [density of primes $1/\log_e m$]
                  $\sqrt{n}/\log\sqrt{n}$ $n^{0.00000001} > \log n$
                  $\sqrt{n}/n^{0.00000001}$ is not helping
        - Fermat Factorization
            - Method: for i = 0,1,2… terminate if $n + i^2 = x^2$
            - Analysis: n,a,b odd, set $i = (b-a)/2$
            - RSA prime choosing lesson: do not take a,b to be too close
        - Exponent Factorization
            - Thm: $x^2 = y^2 \bmod n$, if $x \ne y \bmod n$ and $x \ne y \bmod n$, then $\gcd(x-y,n)$ nontrivial factor
            - Method :
                - Express $k = 2^s *b$ (b odd integer)

            - mu_0 = a^b mod n, for i = 1...s, mu_i = mu_(i-1)^2
            - if mu_(j-1) != -1, last mu that is not 1
            - gcd(mu_(j-1)-1, n) is a non trivial factor also gcd(mu_(j-1)+1, n)
        - Analysis: hope happens
        - Use Exponent Factorization to factor n into p,q in RSA
            - Method: ed-1 = j*phi(n)
            - ½ fail, ½^l fail
            - ed poly in n
    - P-1 Method
        - Method: 2^(B!) = ((2^2)^3)^4… if gcd(b-1, n) > 1, then gcd(b-1, n) is nontrivial factor
        - Analysis: Suppose p-1 has small primes in its prime decomp, so p-1|B!, suppose q-1!|B!, 2^(B!) = 2^(p-1*(B!/(p-1))) = 1 mod p, p|b-1, but q!|b-1, n: pq,p,q,1 but b-1 doesn't have pq as factor, then gcd(b-1,n) = p
        - Lesson: Do not choose p if (p-1) is just small primes in its prime factorization, do not choose q if (q-1) is just small primes in its prime factorization
    - Quadratic Sieve
        - Given odd int to factor, if x^2 cong y^2 mod n, x != +-y mod n (then n divides x+y or n divides x-y), then nontrivial factor is gcd(x+y,n), gcd(x-y,n) [***some n's factors in x+y, some in x-y]
        - Pick a_i near sqrt(n), sqrt(2n), sqrt(3n) so that a_i^2=const*n+small_integer ⇔ a_i^2 = small_integer mod n
          Hope small_integer is a square
        - Analysis: more columns than rows -> linear dependence(det = 0)
    - Sieve of Eratosthenes
- Generating Large Primes
    - Density(1/log_e n, 1/6log_elog_en)
    - Efficient Testing(O(log_e n) tests will be efficient)
        - Ex. 20log_e n numbers will almost guarantee 20 primes
- Primality Testing (both Fermat and M-R, if not 1 => not prime immediately)
    - Fermat Test: if n prime then a^n-1= 1 mod n
        - Method: randomly choose a, test a^n-1= 1 mod n
        - Analysis: mysterious
        - Odd, composite n is Carmichael number if a^n-1= 1 mod n for all a in Z_n*
            - Ex. 561

        - There are infinitely many Carmichael numbers
    - Miller-Rabin Thm: if n prime, either mu_0 = 1 or mu_i = -1, can filter out some Carmichael numbers!
        - Method: randomly choose l integers, check criterion, tell prime/not
        - Analysis: if prime, both says prime

          If not prime, maybe Fermat says prime but M-R doesn't

          P(M-R wrongfully suggests "prime") <= ¼ => P(wrong) = (¼)^l
    - Silly Primality Testing: randomly choose a in 1,2...n-1, compute gcd(a,n)
- Other
    - Lagrange interpolation scheme
        - Given x1,x2...xk distinct, y1,y2...yk, find P(x) = a_k-1x^(k-1)+...+a_1x+a_0 s.t P(xj) = yj
        - Vandermonde matrix: det(V) = Product(xi-xj) for all i<j, invertible!
            - Existence + Uniqueness
        - Another approach: Li(x) = Product((x-xj)/(xi-xj)) (j!=i) j changes so P(xj) = Sum i to k yiLi(xj)
            - Existence shown above
            - Uniqueness: suppose P(xi)=P'(xi) for i=1...k -> P-P' is poly with <=k-1 degree but k roots => P-P' cong 0
    - Field
        - Def: a set V with two binary operations: * and +
        - Z_p,+,* is a field iff p is prime(all units can find inverses)
    - Secret exchange
        - Pick P(x), a_0=s <- secret
        - pick distinct x1,x2...xw
        - distribute (x1, y1), (x2, y2)...(xw,yw) to w people
        - Any k of them can derive secret together