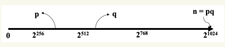
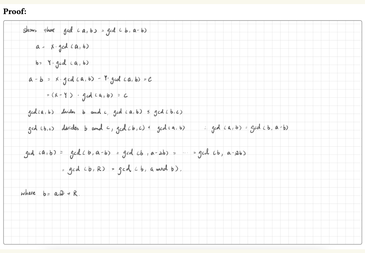
RSA

- * some definition is attached in "math prerequisites" below, I put the content of RSA first to have a first glance of its function
- a public-key crypto-system
- goal
  - (x^e)^d mod N = x — x^e^d = x mod N
  - find N and an encrypt-decrypt pair e and d, when doing this operation, we can get origin plaintext x
- steps
  - key generation
    - pick a co-prime p and q, then calculate N = p * q and φ(n) = (p-1) * (q-1) — e.g. p = 17, q = 23, n = p * q = 391, φ(n) = (p-1) * (q-1) = 235
    - then we pick a prime e which is a co-prime of φ(n) — to simplify the calculation, we choose a small one, e = 3
    - then we could use extended-Euclid to calculate the inverse of e – d, to make e * d = 1 mod φ(n) — a = 3, b = φ(n) = 352
    - by Extended Euclid's, gcd(e, φ(n)) = e * x + φ(n) * y, where x and y is unknown, x is the inverse of e (obviously, it is the d we want), y is the inverse of φ(n), we could omit y for now — 3 * x + 352 * y = gcd(3, 352)
    - we could use the Extended Euclid to deduce the x (aka. the d) and y
      - 1. input (a1, b1) = (φ(n), e) = (352, 3) — quotient = 117, remainder = 1 — 6. return (x1, y1, d1) = (y2, x2 - y2 * ⌊a1 / b1⌋, d1) = (1, 0 - 1 * ⌊352 / 3⌋, d1) = (1, -117,), d1 = a1 * x1 + b1 * y1 = 1
      - 2. input (a2, b2) = (e, φ(n) % e) = (3, 1) — quotient' = 3, remainder' = 0 — 5. return (x2, y2, d2) = (y3, x3 - y3 * ⌊a2/b2⌋, d2) = (0, 1 - 0 * ⌊3 / 1⌋, d2) = (0, 1, 1), d2 = a2 * x2 + b2 * y2 = 1
      - 3. input (a3, b3) = (φ(n) % e, e % φ(n) % e) = (1, 0) — 4. when remainder is 0, return output (x3, y3, d3) = (1, 0, 1), d3 = a3 * x3 + b3 * y3 = 1
      - in all, the x = 1, y = -117
      - 1 * 352 - 117 * 3 = 1 — => 3 * (-117) = 1 mod 352
      - -117 is the inverse of 3 — we prefer to use a positive, so let -117 + 352 = 235
      - 235 is the inverse of 3
      - so d = 235
    - finally we get the pair e = 3, d = 235
  - encryption — e.g. plaintext 73 — y = x^e mod N = 73^3 mod 391 = 389
  - decryption — x = y^d mod N = 389^235 mod 391 = 73
- why it work — Euler's Theorem (see below)
- vulnerability
  - if the adversary could find φ(n), then he can computer secret value d by finding p and q
  - finding φ(n) equals to factoring — **Yes! Consider the equation (X-p)(X-q) = 0, and note that: N - φ(N) + 1 = p+q ⇒ X² – (p+q)X + pq = X² – (N - φ(N) + 1)X + N ⇒ p and q can be found by solving this quadratic equation only if φ(N) is known**
- hard conjecture — when p and q is too large, hard to find prime factorization of n



- math prerequisites
  - Common Divisor — defi.
    - y | x — means x % y = 0, x is evenly divisible by y
    - gcd(c, d) — Greatest Common Divisor of c and d
    - gcd(a, b) = 1 — a and b are relatively prime
    - a = b mod m
      - => m | (a - b)
      - => (a - b) % m = 0
      - => b = a mod m
    - Zn
      - set of integers mod n
      - Zn = { 0, 1, 2, … , n-1 }
    - Z*n
      - set of integers that are relatively prime to n
      - e.g. — Z8 = {0, 1, 2, 3, 4, 5, 6, 7} Z*8 = {1, 3, 5, 7}
  - Euclid's Algorithm
    - for finding gcd(a, b)
    - content
      - gcd(a, b) = gcd(b, r1) = gcd(r1, r2) = … — **Proof:**



      - a = q1 * b + r1 — r1 = a % b
      - b = q2 * r1 + r2
      - r1 = q3 * r2 + r3
      - …
      - rn = q_n+2 * r_n-1 + 0
      - r_n+1 = gcd(a, b)
    - implement — **function** Euclid(a, b)
Input: Two integers a and b, where $a \geq b \geq 0$
Output: gcd(a, b)
1. **if** $b = 0$:
2. **return** a
3. **return** Euclid($b, a \bmod b$)
  - Extended Euclid's Algorithm
    - efficient way to find inverse — given d, a and b, we can use this algo. to efficiently calculate the coefficient x, y in d = x * a + y * b, and we can verify if d = x * a + y * b is true — where d is supposed to be the gcd(a, b)
    - content
      - r1 = a - q1 * b
      - r2 = b - q2 * r1 — = (-q2) * a + (q1 * q2 + 1) * b
      - …
      - ri = (…) a + (…) b
    - implement — **function** extended-Euclid(a, b)
Input: Two integers a and b, where $a \geq b \geq 0$
Output: Integers (x, y, d) such that $d = gcd(a, b)$ and $ax + by = d$
1. **if** $b = 0$:
2. **return** $(1, 0, a)$
3. $(x', y', d) =$ extended-Euclid($b, a \bmod b$)
4. **return** $(y', x' - \lfloor a/b \rfloor y', d)$

    - calculation example:
      - when input a = 25, N = 11
        1. On the first call, $a = 25$ and $b = 11$. Since $b \neq 0$, the function will make a recursive call with b and $a \mod b$, which is 11 and 3 respectively.
        2. On the second call, $a = 11$ and $b = 3$. Again, $b \neq 0$, so the function will make another recursive call with b and $a \mod b$, which is 3 and 2 respectively.
        3. On the third call, $a = 3$ and $b = 2$. The function will make yet another recursive call with 2 and 1.
        4. On the fourth call, $a = 2$ and $b = 1$. Now, with another recursive call, the values become $a = 1$ and $b = 0$.
        5. On this fifth call with $b = 0$, the function returns $(1, 0, 1)$ as $\gcd(1, 0) = 1$.

        From the fourth call:
        $(x', y', d) = (1, 0, 1)$
        Returning:
        $(y', x' - \lfloor \frac{a}{b} \rfloor y') = (0, 1 - \lfloor \frac{2}{1} \rfloor 0) = (0, 1)$

        From the third call:
        $(x', y', d) = (0, 1, 1)$
        Returning:
        $(y', x' - \lfloor \frac{3}{2} \rfloor y') = (1, 0 - 1) = (1, -1)$

        From the second call:
        $(x', y', d) = (1, -1, 1)$
        Returning:
        $(y', x' - \lfloor \frac{11}{3} \rfloor y') = (-1, 1 - (-3)) = (-1, 4$

        From the first call:
        $(x', y', d) = (-1, 4, 1)$
        Returning:
        $(y', x' - \lfloor \frac{25}{11} \rfloor y') = (4, -1 - 2) = (4, -3)$

        So, for $a = 25$ and $b = 11$, the values are:
        $x = 4$, $y = -3$, and $d = \gcd(25, 11) = 1$

        Thus, the result is $x = 4$, $y = -3$, and $d = 1$.
  - Bazout's theorem
    - exist x, y such that gcd(a, b) = α * a + β * b, where α and β could be negative — β * b = 1 mod a
    - defi. of inverse
      - β is the inverse of b mod a — β * b = 1 mod a
      - α is the inverse of a mod b — α * a = 1 mod b
      - * there is at most 1 inverse x in Zn for x * a = 1 mod n
        - no inverse case — $2k \neq 1 \pmod 4$ $\forall k, \ldots, \ldots$ — a = 2, n = 4
        - when inverse exists? — $ax \equiv 1 \pmod b \iff a$ and b coprime $\iff ax + by = 1$ $\iff \gcd(a, b) = 1$ — that is to say, when a and N are co-prime, the inverse exists
  - Euler's Totient Function
    - φ(n) — = |Z*n| — count of the elements in it — e.g. — φ(8) = 4
    - properties
      - the number of positive integers that are relatively prime to n and less than n
      - p is prime, φ(p) = p-1
      - p, q are distinct primes, n = p * q, |Zpq| = p * q
      - p, q are distinct primes, n = p * q, φ(n) = (p - 1) * (q - 1)
  - Euler's Theorem
    - used to easily reduce large powers modulo
    - content
      - for all a ∈ Z*n, a^{φ(n)} = 1 mod n
      - or, for all a ∈ Z*n and k ≥ 0, a^{k * φ(n) + 1} = a mod n
    - proof — https://en.wikipedia.org/wiki/Euler%27s_theorem — [haven't figure it out yet]
    - generalization
  - RSA Theorem
    - If p, q are distinct primes, n = p * q,
    - for all a ∈ Zn, a ^ φ(n) = 1 mod n
    - or, for all a ∈ Zn, and k ≥ 0, a ^ (k * φ(n) + 1) = a mod n.