Rsa has two keys private key and public key. You sign something using public key and verify the signature using private key.

(e,n)public key (d)private key

N is a very large number 2000-4000 bits long, e is usually 65537 not a secret

N = p.q generate two random values p and q to multiply it and generate a random number n

Breaking rsa has given e and n

Factor n into p and q

We use euler totient function to calculate the totient of n

φ(n) = (p-1).(q-1)

e.d ≅ 1 mod φ(n) is congruent to

now we know that n is a composite number because it produces two prime numbers when prime factorization is conducted

ferments factorization algorithm is effective only when the prime numbers p and q are not very fifferent from each other

n = (a^2 – b^2) = (a+b) (a-b)

b^2 = a^2 – N

For this to work we must find the value of b as a squared number to balance the equation above

square root of N is where we want to start and move slowly up through a to find a plausible value for b

for this we use a ceiling function.

Initial guess of a would be a = square root of N (integer right above it)

In order for this to work, we add 1 to a to find number that are b squared(going to the next integer above it)

what this equation basically means is we want to find a number d, which when we multiply by n will give us an intermediate value than can be reduced by mod φ(n) which will give is 1.

n = 5261933844650100908430030083398098838688018147149529533465444719385566864605781576487305356717074882505882701585297765789323726258356035692769897420620858774763694117634408028918270394852404169072671551096321238430993811080749636806153881798472848720411673994908247486124703888115308603904735959457057925225503197625820670522050494196703154086316062123787934777520599894745147260327060174336101658295022275013051816321617046927321006322752178354002696596328204277122466231388232487691224076847557856202947748540263791767128195927179588238799470987669558119422552470505956858217654904628177286026365989987106877656917

random number

n.nbits()

a = isqrt(n) + 1

a

while True:

....: b2 = a^2 - n

....: if is\_square(b2):

....: b = sqrt(b2)

....: break

....: a = a + 1

a

b

p = a + b

q = a – b

e = 65537

phi\_n = (p-1)\* (q-1)

d = inverse\_mod(e,phi\_n)