Rsa has two keys private key and public key. You sign something using public key and verify the signature using private key. Fermat's algorithm can be used in classical computer to find the private key from public key of RSA when keys are chosen very close to each other and not in random. However, this is relevant to post-quantum cryptography since quantum computers can easily use brute force since it has powerful computational ability.

(e,n)public key

(d)private key

N is a very large number 2000-4000 bits long

e is usually 65537 not a secret (public key)

N = p.q , generate two random prime values p and q to multiply it and generate a random composite 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

i.e. $\phi(n) = (p-1).(q-1)$

given, $e.d \cong 1 \bmod \phi(n)$ 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 $\phi(n)$ which will give is 1.

Code:

n =

5261933844650100908430030083398098838688018147149529533465444719385566864605781
5764873053567170748825058827015852977657893237262583560356927698974206208587747
6369411763440802891827039485240416907267155109632123843099381108074963680615388
1798472848720411673994908247486124703888115308603904735959457057925225503197625
8206705220504941967031540863160621237879347775205998947451472603270601743361016
5829502227501305181632161704692732100632275217835400269659632820427712246623138
8232487691224076847557856202947748540263791767128195927179588238799470987669558
1194225524705059568582176549046281772860263659899871068776569 17

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)