

Some attacks against RSA

Jean-Sébastien Coron

Université du Luxembourg

1 Coppersmith Attack on partially known message encryption with RSA

1.1 SAGE

Download and install the Sage library [1].

1.2 Finding small root of a modular polynomial equation of degree 2

The following code generates an RSA key with a modulus N of n bits, generates a random polynomial:

$$f(x) = x^2 + ax + b \pmod{N}$$

with a small root $|x_0| < 2^{n/3}$, and recovers this root using Coppersmith's technique.

```
def keyGen(n=256):
    "Generates an RSA key"
    while True:
        p=random_prime(2^(n//2));q=random_prime(2^(n//2));e=3
        if gcd(e,(p-1)*(q-1))!=1: break
    d=inverse_mod(e,(p-1)*(q-1))
    Nn=p*q
    print("p=",p,"q=",q)
    print("N=",Nn)
    print("Size of N:",Nn.nbits())
    return Nn,p,q,e,d

"Finds a small root of polynomial x^2+ax+b=0 mod N"
def CopPolyDeg2(a,b,Nn):
    n=Nn.nbits()
    X=2^(n//3-3)
    M=matrix(ZZ,[[X^2,a*X,b],\
                 [0 ,Nn*X,0],\
                 [0 ,0 ,Nn]])
    V=M.LLL()
    v=V[0]
    R.<x> = ZZ[]
    f=sum(v[i]*x^(2-i)/X^(2-i) for i in range(3))
    return f.roots()

def test():
    ""Generates a random polynomial with a small root x0 modulo Nn
    and recovers his root.""
    Nn,p,q,e,d=keyGen()
    n=Nn.nbits()
    x0=ZZ.random_element(2^(n//3-10))
    a=ZZ.random_element(Nn)
```

```

b=mod(-x0^2-a*x0,Nn)
print("x0=",x0)
print(CopPolyDeg2(a,b,Nn))

```

You can also evaluate the code on <https://sagecell.sagemath.org/?q=ggjafk>

1.3 Polynomials of degree 3

Modify the previous code to find small roots of polynomials of degree 3. What is the new bound on x_0 ?

1.4 Application to breaking RSA encryption with small exponent e

Let

```

N = 70934618878370780806798506702845692019844344515094386800753974870084300
8346940940287222002891029537499396908056848095671838771377752278390679951948
8084757074500737145779301525290570571375547245377199125154867124084225417111594
69388603174949214558211229232753033853223644282969544939436208933732047182752629

```

be an RSA modulus of size 1017 bits. Let m be a message with $m < 2^{168}$. Let

$$c = (2^{1016} + m)^3 \pmod{N}$$

with

```

c = 106931790228306983984681105468768847489271870726744288161524001625441564428928138
08276484754633045564572102067212938356933286498024763104799136897507623550810216703412
812182136692774490788261792897332208158526227545478955914179713569378150960980188137695
1044409238368040773136628352506077437800520604807524

```

Recover the message m using Coppersmith's technique.

2 Optional: Factoring with high order bits known

Consider the following 1023-bit RSA modulus $N = pq$ with:

```

N =604889055931441335818271295619602228995645508868580059819472692573987079478494107
585591266723162484945555801294297572527393540117546661252625861419705626992557920488818
059965226909344028835156910032162272321210846317964685824843613238180415223483321047850
00833095087675938985207931444661998445034706874951507

```

The secret prime p is a 512-bit integer, and we are given its high-order 347 bits. More precisely, we write $p = P + x_0$ with $P \pmod{2^{165}} = 0$ and $0 \leq x_0 < 2^{165}$, with:

```

P =8097223648343011172606583468066750480111622020173971387518911659260792395887115626895
162782768100202412325404414825940145649070725756396542410376064532480

```

The goal is to recover the prime factor p using the technique described in the slides.

3 Optional: Wiener's attack against $d < N^{1/4}$

1. Implement the Extended Euclidean algorithm to compute a sequence of integers a_i, b_i such that $a_i \cdot u \equiv b_i \pmod{e}$, given as input u and e .
2. Write a function that given e , generates $u = b/a \pmod{e}$ with $\gcd(a, e) = 1$, $\gcd(a, b) = 1$, $0 \leq a < e^{1/4}$ and $0 \leq b < e^{3/4}/2$
3. Modify the Extended Euclidean algorithm to recover a, b given u and e .
4. Let

```
N = 10130504119574269789293062680493199194915552096732359153857412158756437203
    47135740015147283114417898552525558307523182360246570149631744078935953061839
    8999802968854212580408518026278696751955766440302916899698207562116802903505
    967192902681681626273432110182016329136690925805959479917572582616951641454736961
```

be an RSA modulus, and

```
e = 3134657917572805640643483213467129157665628760339358509398392813132282456358
    9978332807232269972876643175705237878721777376331255131580937895216938815784334
    7877922918358552412126143406736881542173183213899346722202070378656735309988244
    718851660019687049019311508591439699717362940974454351066233999128990389
```

be a public exponent, with $d < N^{1/4}$. Recover the private exponent d using Wiener's attack.

References

1. Sage Mathematical Library, Available at <http://www.sagemath.org/>