# Some attacks against RSA

Jean-Sébastien Coron

University of Luxembourg

## 1 Coppersmith Attack on partially known message encryption with RSA

Install the Sage library, available at `http://www.sagemath.org/`. Alternatively, you can use the Sage Cell Server at `https://sagecell.sagemath.org`. Please submit a `.ipynb` file.

### 1.1 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.

```
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,"\nq=",q,"\nN=",Nn,"\nSize of N:",Nn.nbits())
    return Nn,p,q,e,d
```

The following code generates a random polynomial:

$$f(x) = x^2 + ax + b \mod N$$

with a small root $|x_0| < 2^{n/3}$

```
def genPoly(Nn):
    "Generates a polynomial with a small root modulo Nn"
    n=Nn.nbits()
    x0=ZZ.random_element(2^(n//3-10))
    a=ZZ.random_element(Nn)
    b=mod(-x0^2-a*x0,Nn)
    return x0,a,b
```

The following code recovers this root using Coppersmith's technique.

```
"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)
    X=ZZ(round(Nn^(1/3)/6))
    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()
    x0,a,b=genPoly(Nn)
    print("x0=",x0)
    print(CopPolyDeg2(a,b,Nn))
```

### 1.2 Polynomials of degree 3

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

### 1.3 Application to breaking RSA encryption with small exponent $e$

Let

$$N = 142639371614095202938903608951283735078993536755699322233976633286631773719084324844378653059811664547922971205612133227293728494568499205201144787388804821468650077465401658439412828003454774148332616313216897519024187861902815466489320442211861711728708981721748268801907751102500134758552327889466753559787$$

be an RSA modulus of size 1024 bits. Let $m$ be a message with $m < 2^{170}$. Let $c = (2^{1023} + m)^3 \mod N$, with

$$c = 113807927628281006288302112046419141997139505696892267606920151428506610503869463576430406063715418078143540635697474778928769599103592191445791703245051109236305604506698946585086432715949275265685118224544838597491448131611348156424314208898744945110722766428735407595892966462872377115663466689994813590934$$

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:

$$N = 604889055931441335818271295619602228995645508868580059819472692573987079478494107585591266723162484945555801294297572527393540117546661252625861419705626992557920488818059965226909344028835156910032162272321210846317964685824843613238180415223483321047850008330950876759389852079314446619984450347068749515 07$$

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

$$P = 809722364834301117260658346806675048011162202017397138751891165926079239588711562689516278276810020241232540441482594014564907072575639654241037606453248 0$$

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

$$N = 101305041195742697892930626804931991949155520967323591538574121587564372034713574001514728311441789855252555830752318236024657014963174407893595306183989980296885421258040851802627869675195576644030291689969820756211680290350596719292026816816262734321101820163291366909258059594799175725826169516414547369614$$

be an RSA modulus, and

$$e = 31346579175728056406434832134671291576656287603393585093983928131322824563589978332807232269972876643175705237878721777376331255131580937895216938815784334787792291835855241212614340673688154217318321389934672220207037865673530998824471885166001968704901931150859143969971736294097445435106623399912899038 9$$

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/`