TSINGHUA UNIVERSITY, INSTITUTE FOR INTERDISCIPLINARY INFORMATION SCIENCES

Finding Modular e-th root and attacks for the RSA system

Gao Minbo, Xie Tianle

January 10, 2020

Abstract

In this report we study the approach of getting modular e-th root and two different paradigms of attacks for the RSA cryptographic system.[1] First goal is to discuss the finding of modular e-th root when the factorization of N is known. In detail, we divide the situation into several different cases: $e = 2$, general e for the prime p, and for N is a composite whose factorization is known. Our second goal is to find algorithms that factor an RSA modulus $N = pq$ in time polynomial in the bit-size of N. We assume p and q are primes of the same bit length, and under this assumption, we discuss the famous Wiener's attack. In the case of $d < \frac{N^{0.25}}{3}$, both continued fraction and lattice method are introduced for implementing Wiener's attack. We also include some of the advanced results got by the lattice paradigms.

Key words: modular e-th root, RSA cryptographic system, Wiener's attack, continued fraction, lattice based cryptographic attack.

CONTENTS

1 FINDING MODULAR e-TH ROOT

Person in Charge: Xie Tianle

1.1 PRELIMINARY

We will first define the problem formally. Given integers e, c, and N, we want to find all integers x satisfy that

$$x^e \equiv c \mod N.$$

More precisely, we will only consider the situation when we have already known the factorization of N in this part. We will first show the algorithm when $e = 2$ and N is a prime, then we can generalize these techniques to more complex situations.

1.2 FIND e-TH ROOT WHEN N IS A PRIME

1.2.1 CASE $e = 2$

The instructor has shown the Cipolla's algorithms in class. For the equation

$$x^2 \equiv c \mod p,$$

- Find integer a such that $a^2 - c$ is not a quadratic residue.
- Let $\omega = \sqrt{a^2 - c}$.
- $x = \pm(a-\omega)^{(p+1)/2}$ is the two root of the equation above.

We will use the primitive root of p to prove the assertion.[2] Recall the definition of primitive root: r is a primitive root of integer N means that $\Phi(N)$ is the smallest integer k such that

$$r^k \equiv 1 \mod N.$$

When N is a prime p, this implies that the $\frac{p-1}{2}$ quadratic residues are exactly $r^2, r^4, \cdots, r^{p-1}$. By $r^{(p-1)/2} \equiv -1 \mod p$ we know that

$$\omega^p = (a^2 - c)^{(p-1)/2} \equiv -1 \mod p.$$

Now we can use these characteristic to simplify the formulas above:

$$\begin{aligned}
& x^2 \\
=& (a-\omega)^(p+1) \\
\equiv& (a-\omega)\cdot(a^p - \omega^p + p*K) \mod p \\
\equiv& (a-\omega)\cdot(a-(-1)*\omega) \mod P \\
\equiv& a^2 - \omega^2 \mod P \\
=& c
\end{aligned}$$

Notice that $x^2 \equiv c \mod p$ has 2 roots and all of them should be integers. I this way, though $\pm(a-\omega)^{(p+1)/2}$ need some calculation outside integers, the final output must be a integer.

At last, we need to verify that find satisfying a is easy. Consider the following quadratic equation:

$$x^2 - 2ax + c = 0.$$

The two roots are $a \pm \sqrt{a^2 - c}$. So for constant c, the number of unsatisfactory a is exactly the number of those quadratic equation with two integer roots. By Vieta theoremïijN̈ we know the two roots x_1, x_2 satisfy that $x_1 \cdot x_2 = c$, so there are $(p+1)/2$ unsatisfactory equations, thus we can find satisfactory a with probability $(p-1)/2p$, nearly $\frac{1}{2}$.

1.2.2 GENERIC e

Using the technique above, we can derive the following algorithms for generic e and prime p: For the equation

$$x^e \equiv c \mod p,$$

- Find integer a such that $a^e - c$ is not a e-th residue.
- Let $\omega = (a^e - c)^{(1/e)}$.
- $x = (a-\omega)^{(1+p+p^2+\cdots+p^{e-1})/e}$ is one root of the equation above.

Similarly, we can find the e roots of equation $x^e - \omega^e$ and then get the decomposition directly. First, $x = \omega$ is obviously a root of the equation. Then by

$$\begin{aligned}(\omega^p)^e \equiv & (\omega^e)^p \mod p \\ \equiv & (a^e - c)^p \mod p \\ \equiv & a^e - c \mod p \\ \equiv & \omega^e \mod p\end{aligned}$$

we know that w^p is also a root of the equation. In this way, we know the e roots are $w, w^p, w^{p^2}, \cdots, w^{p^{e-1}}$. So

$$\begin{aligned}& x^e \\ =& (a-\omega)^{(1+p+p^2+\cdots+p^{e-1}))} \\ \equiv & (a-\omega)\cdot(a-\omega)^p\cdot(a-\omega)^{p^2}\cdots(a-\omega)^{p^{e-1}} \mod p \\ \equiv & (a-\omega)\cdot(a^p-\omega^p)\cdots(a^{p^{e-1}}-\omega^{p^{e-1}}) \mod P \\ \equiv & a^e - \omega^e \mod P \\ =& c\end{aligned}$$

Now we verify some details in this algorithm. Notice that we need to calculate a power with index $(1+p+p^2+\cdots+p^{e-1}))/e$, we will show that we can assume this number is an integer.

The basic case is when e and $p-1$ are coprime. By Bezout's identity, we can find integers u, v such that $ue - v(p-1) = 1$.

$$c^u \equiv x^{ue} \equiv x^{v(p-1)+1} \equiv x \mod p.$$

In this way, we can easily find the only root c^u.

Moreover, for composite $e = qe_0$, we can first find all q-th roots, then find the corresponding e_0-th roots of the roots above. Therefore, we can focus on the situation when $e|p-1$. In this way,

$$(1 + p + p^2 + \cdots + p^{e-1})) \equiv (1 + 1 + 1 + \cdots + 1) \equiv 0 \mod e.$$

1.3 Find e-th root when N is a composite, and factorization is known

Notice that for a generic composite N, we can use Chinese remainder theorem to reduce it into the simple case when N is a power of prime.

We will use $N = p^2$ as an example to demonstrate the algorithm. Assume we have already solve the following equation:

$$x_0^e \equiv c \mod p.$$

Consider the following p integers $x_0, x_0 + p, x_0 + 2p, \cdots, x_0 + (p-1)p$. Notice for integer i,

$$(x_0 + ip)^e \equiv x_0^e + eip + Cp^2 \equiv x_0^e + eip \mod p,$$

where C is an integer. Thus we know there must be one i satisfies that $x_0 + ip$ is the root of equation $x_0^e \equiv c \mod p$ because we can assume $\gcd(e, p) = 1$ by Fermat Theorem.

Particularly, when $p = 2$ is the only even prime, we need to change some small point in the algorithm, further analysis can be found in [4].

1.4 Complexity analysis of the above algorithms

For all the algorithms above, we only need to calculate a power of a particular number with index less than $O(p^e)$. Thus when we know the factorization of N, we can find modular e-th root with time complexity $O(poly(e, \log p))$ with high success rate.

2 ATTACKS ON RSA: AN INTRODUCTION TO THE LATTICE METHOD

Person in Charge: Gao Minbo

2.1 INTRODUCTION TO THE WIENER'S RSA ATTACK: NOTATIONS AND ASSUMPTIONS

2.1.1 NOTATIONS FOR THE RSA PROBLEM

First of all, let's make notations clear, which will be frequently used in the later part in this report. Recall that for an RSA system, we have the modular N and exponent e, both are integers. For the modular $N = pq$, we know p and q are distinct primes (of the same length, in practice). So by definition, we can get $\varphi(N) = (p-1)(q-1) = N-(p+q)+1$ The number pair (N,e) is the public key. The secret key d satisfies the equation:

$$ed - k\varphi(N) = 1. \tag{2.1}$$

This equation will be frequently used throughout this report.

2.1.2 USEFUL BOUNDS FOR ESTIMATING THE PARAMETERS WHEN $d \leq N^{0.25}$

In this section, we derive some useful bounds for the parameters mentioned above. In some of the derivations, we use the assumption that $d \leq N^{0.25}$. First, without generality, we can assume $p < q$. So the following inequality clearly holds:

$$p < \sqrt{N} < q < 2p < 2\sqrt{N}. \tag{2.2}$$

Setting $c = \sqrt{\frac{q}{p}}$, we know that $1 < c < \sqrt{2}$. And by this assumption, we have: $p+q = (c+\frac{1}{c})\sqrt{N}$. By the monotone property of the function $f(x) = x + \frac{1}{x}$ when $x \geq 1$, we know a more dedicated bound holds:

$$2\sqrt{N} < p+q < \frac{3}{\sqrt{2}}. \tag{2.3}$$

By noticing that $\sqrt{N}$ has the same bit length as p, it is clear that:

$$q - p < \sqrt{N}. \tag{2.4}$$

Furthermore, the following equality holds by definition:

$$\frac{N}{2} < \varphi(N) < N. \tag{2.5}$$

2.2 THE WIENER'S ATTACK FOR THE RSA IN THE FORM OF CONTINUED FRACTIONS

2.2.1 WIENER'S THEOREM FOR ATTACKING THE RSA

We state the theorem as follows:

Theorem 1. *Let $N = pq$ with $p < q < 2p$, $d < \frac{1}{3}N^{0.25}$. Given (N, e) with $ed - k\varphi(N) = 1$, the attacker can efficiently recover d.*

Now the proof is as follows:

Let $G = \gcd(p-1, q-1)$, $\lambda(N) = \mathrm{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{G} = \frac{\varphi(N)}{G}$, $k = \frac{K}{\gcd(K,G)}$ and $g = \frac{G}{\gcd(K,G)}$. It is clear that, for some integer k, we have: $\left|\frac{e}{\lambda(N)} - \frac{k}{d}\right| = \frac{1}{d\lambda(N)}$.

Using the bounds above, we obtain:

$$\begin{aligned} \left|\frac{e}{N} - \frac{k}{Gd}\right| &= \left|\frac{edG - kN}{NGd}\right| \\ &= \left|\frac{edG - k\varphi(N) - kN + k\varphi(N)}{NGd}\right| \\ &= \left|\frac{1 - k(N - \varphi(N))}{NGd}\right| \\ \leq \left|\frac{3k\sqrt{N}}{NGd}\right| &= \frac{3k\sqrt{N}}{\sqrt{N}\sqrt{N}Gd} \leq \frac{3k}{d\sqrt{N}} \end{aligned} \tag{2.6}$$

By using the bound of $k < d$ and $d < \frac{N^{0.25}}{3}$, we finally get:

$$\left|\frac{e}{N} - \frac{k}{Gd}\right| \leq \frac{3k}{d\sqrt{N}} < \frac{1}{d \cdot 2d} = \frac{1}{2d^2} \tag{2.7}$$

which means we can just compute the continue fraction expansion of $\frac{e}{N}$ to find d, simply by testing the relation whether $2^{ed} \equiv 1(\mod n)$ for the denominator.

2.3 The Wiener's attack for the RSA in the form of lattice

2.3.1 Integer lattice and LLL algorithm

Now, to extend Wiener's attack in a more systematic way, we use the form of lattice to present the above result and show these can be extended in a more general way.[5] Formally, a lattice is a module over a ring which is embedded in a vector space over a field. For our use, we just need the usual $Z-$module over $\mathbb{R}^n$. So we simply define the lattice as follows: For $v_1, v_2, \cdots, v_n \in \mathbb{Z}^m, m \geq n$ be linear independent vectors. The lattice L spanned by these vectors is defined by:

$$L = \left\{ v \in \mathbb{Z}^m | v = \sum_{i=1}^{n} a_i v_i \text{ with } a_i \in \mathbb{Z} \right\} \tag{2.8}$$

By applying the Gram-Schmidt orthogonalization to the basis vectors, we get a new basis $b_1^*, b_2^*, \cdots, b_n^*$. The determinant of the lattice L is defined as:

$$\det(L) = \prod_i ||b_i^*||. \tag{2.9}$$

Algorithm 1 LLL lattice basis reduction algorithm

Perform Gram-Schmidt, but do not normalize:
$ortho$:= gramSchmidt($\{b_0,\ldots,b_n\}$) = $\{b_0^*,\ldots,b_n^*\}$
Define $\mu_{i,j} = \frac{\left(b_i,b_j^*\right)}{\left(b_j^*,b_j^*\right)}$, which must always use the most current values of b_i, b_j^*.
$k = 1$.
while k$\leq n$ **do**
  for j from $k-1$ to 0 **do**
    if $\left|\mu_{k,j}\right| > \frac{1}{2}$ **then**
      $b_k = b_k - \lfloor \mu_{k,j} \rceil\, b_j$
      Update ortho-entries and related $\mu_{i,j}$'s as needed.
  if $\left\langle b_k^*, b_k^* \right\rangle \geq \left(\delta - \left(\mu_{k,k-1}\right)^2\right)\left\langle b_{k-1}^*, b_{k-1}^* \right\rangle$ **then**
    $k = k+1$
  else
    Swap b_k and b_{k-1}.
    Update ortho entries and related $\mu_{i,j}$'s as needed.
    $k = \max(k-1, 1)$

Now we introduce the famous LLL algorithm, which can compute the reduced basis in polynomial time. The LLL algorithm is as follows:

It can be shown that the running time of the algorithm is $O\left(d^5 n \log^3 B\right)$ for d vectors basis lattice and $B = \max\left(\|\mathbf{b_1}\|_2, \|\mathbf{b}_2\|_2, \ldots, \|\mathbf{b}_d\|_2\right)$.

For the convenience of analysis, we introduce the following result which is a corollary of Lenstra, Lenstra and Lovasz's original result.

Theorem 2. *Let $L \in \mathbb{Z}^n$ be a lattice spanned by $B = \{b_1,\ldots,b_n\}$. The L3-algorithm outputs a reduced lattice basis $\{v_1,\ldots,v_n\}$ with*

$$\|v_i\| \leq 2^{\frac{n(n-1)}{4(n-i+1)}} \det(L)^{\frac{1}{n-i+1}} \quad \text{for } i = 1,\ldots,n. \tag{2.10}$$

in time polynomial in n and in the bit-size of the entries of the basis matrix B.

Notice that, in particular, this theorem can derive the famous theorem by Minkowski:

Theorem 3. *Every n-dimensional lattice L contains a non-zero vector v with $\|v\| \leq \sqrt{n}\det(L)^{\frac{1}{n}}$.*

2.3.2 Wiener's attack in the form of lattice

Now we can reformulate the Wiener's attack in a more compact way. The following theorem is of great importance in this method:

Theorem 4. *Consider a polynomial $f(x,y)$ which satisfies the following condition:*
(i) $f(x_0,y_0) \equiv 0 \mod N$, where $|x_0| \leq X, \left|y_0\right| \leq Y$.
(ii) $\left\|f\left(xX, yY\right)\right\|_1 < \frac{N}{\sqrt{2}}$
Then $f(x_0,y_0) = 0$ holds on $\mathbb{Z}$

The proof of this theorem is as follows:

$$|f(x_0, y_0)| \leq \sum_i |a_i||x_0|^i |y_0|^i < N. \tag{2.11}$$

The last equation holds by the change of the $L-1$ norm to $L-2$ norm and the second condition.

Now we consider the attack: Let's define $f(x,y) := ex+y$. Then we know $f(d, k(p+q)-1) \equiv 0 \mod n$ by the RSA requirement. In addition $g(x,y) := Nx \equiv 0 \mod n$ always holds. So we can consider the lattice spanned by the vector set $(eX, Y), (NX, 0)$. By applying the LLL algorithm theorem, we know we can find a function $h(x,y)$ with the same root, by integer combination of the above two functions, such that:

$$||h(xX, yY)|| \leq \sqrt{2NXY} \leq \sqrt{\sqrt{2}N^2/3} < \frac{N}{\sqrt{2}}. \tag{2.12}$$

So by the above theorem, we know it suffices to find roots on $\mathbb{Q}$ for the function $h(x,y)$. Notice that h is linear for the two varibles, we know that finding the root of h is direct. In other words, we've proved the Wiener's attack theorem via lattice method.

2.4 More results of RSA attacks using lattice method

From above, we introduce two variants of the Wiener's attack. It is rather tempting to say that continued fraction method is simple and straight. However, the lattice method can be generalized and used to prove some more strong results[3], some of which are included in the following:

Theorem 5. *Given an RSA public key tuple (N,e), where $N = pq$. Suppose that e satisfies an equation $ew + z \equiv 0 \mod \varphi(N)$ with*

$$0 < w \leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}}\frac{N^{\frac{3}{4}}}{p-q} \quad \text{and} \quad |z| \leq \frac{1}{8}\frac{e}{\phi(N)}\frac{p-q}{N^{\frac{1}{4}}} \cdot w \tag{2.13}$$

Then N can be factored in time $O(log^2(N))$.

Theorem 6. *Given an RSA public key tuple (N,e) with $N = pq$ and secret exponent d. Let $q < N^\beta$ and $d_p \leq N^{\frac{1-3\beta+\beta^2}{2}} - \epsilon$. Then N can be factored in time $O(log^2(N))$.*

Theorem 7. *For every fixed $\delta > 0$ there exists an integer N_0 such that for every $N > N_0$, the following holds: Given an RSA public key tuple (N,e) with $N = pq$ and secret exponent d. Let $q < N^\beta$ and $d_p \leq N^{1-\frac{2}{3}(\beta+\sqrt{3\beta+\beta^2})-\epsilon}$ Then we can find the factorization of N in time polynomial in log(N).*

References

[1] D. Boneh and V. Shoup. A graduate course in applied cryptography. *Draft 0.2*, 2015.

[2] C. P. Chengdong Pan. *Elementary number theory*. Peking University Publisher, 2013.

[3] A. Dujella. Continued fractions and rsa with small secret exponent. *arXiv preprint cs/0402052*, 2004.

[4] C. J. Kefan Dong. Modular eth root in rsa. *Crypto project*, 2017.

[5] A. May. *New RSA vulnerabilities using lattice reduction methods*. PhD thesis, University of Paderborn Paderborn, 2003.