


ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

A study of the Euclidean attack on RSA

ΦΛΩΡΙΑΣ ΠΑΠΑΔΟΠΟΥΛΟΣ (Α.Ε.Μ.: 874)

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΔΗΜΗΤΡΙΟΣ ΠΟΥΛΑΚΗΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2023

Abstract

This study was made for an assignment of the class "Cryptography" of my master's program. For the assignment, the paper "An application of Euclidean algorithm in cryptanalysis of RSA" [1], had to be studied, analyzed and represented in a more explanatory and student-friendly way.

This "report" will consist of two chapters, one theoretical in which the main concepts of the paper above will be represented thoroughly, and one more practical in which the EUCLID-ATTACK algorithm (proposed in [1]) will be explained and implemented. Moreover, concerning the implementation, it was done using Python and creating an executable program which this "report" accompanies. This executable will be used in order to create some toy examples of application of the algorithm.

Keywords: RSA, attack on RSA, small private exponent, Euclidean algorithm, EUCLID-ATTACK

Contents

Chapter 1

Theoretical survey

As mentioned before, in this chapter the theory behind the application of the Euclidean algorithm on the cryptanalysis of RSA will be explained at length. But, before presenting the how and the why it works, an introduction will be made, mentioning the results of some relative papers, as well as the background and relative conditions under which the attack presented works.

1.1 Introduction

Firstly, we assume that the setup of the classic RSA cryptosystem is known to the reader but mention it for completeness:
"Let p and q be two primes, > 2, of the same size ($l(p) = l(q) = l$) and set $n = pq$. We consider the key space $K = \{k \in \mathbb{Z} \,|\, 1 < k < \phi(n)\ \&\ gcd(k, \phi(n))\}$ of RSA and choose $e, d \in K$ such that

$$ed \equiv 1 \,(mod\, \phi(n)) \tag{1.1}$$

Then, we define the public and private keys of RSA to be (n, e) and d, respectively. Moreover, the encryption and decryption functions respectively are

$$E_e : \mathbb{Z}_n \to \mathbb{Z}_n,\ x \mapsto x^e \,mod\, n$$
$$E_d : \mathbb{Z}_n \to \mathbb{Z}_n,\ x \mapsto x^d \,mod\, n"$$

It is apparent that the different values that e and d can get can affect RSA's properties considerably. For example, if e (or d) is chosen to be small then the encryption (respectively, decryption) process can be faster and less space can be used in order to store the key. In fact, one situation where the use of short exponents is particularly advantageous is when there is a large difference in computing power and available space between two communicating devices, e.g. a smart card and a computer. In this case, as we would like to not minimize the amount of processes done in the smart card, a shorter exponent would be preferable. This sounds like a good idea at first but in reality it is not due to the class of short exponent attacks on RSA.

The most known attack of this kind was first proposed in 1990, by Wiener [2], in a paper where he utilizes an algorithm that used continued fractions to discover a short private key d, in the case $d < n^{1/4}/3$. More precisely, Wiener's thinking starts from the equivalence relation that connects e and d, and performs some calculations until he reaches the equation

$$\frac{e}{pq} = \frac{k}{dg}(1 - \delta), \quad \text{where } \delta = \frac{p + q - 1 - \frac{g}{k}}{pq}$$

with k and g being some integers we won't explain further. We note that the $\frac{e}{pq}$ consists entirely of public information and is a close underestimate of k/dg, if δ is small enough. Wiener's attack then works by generating the continued fraction expansion of e/pq until its close underestimate k/dg can be found using his continued fraction algorithm. This algorithm can be successful only

if δ is small enough, a condition which can be proved to be equivalent to $d < n^{1/4}/3$. Finally, about its computation speed, it can be proven that the overall time complexity of Wiener's attack is $\mathcal{O}((\log n)^3)$ bit operations.

Wiener's attack has been generalized multiple times and extended so that d can be broken for a few more bits in [3, 4]. Also, in [5], results from lattice theory were used to present a variation of Wiener's attack that can compute n in time $\mathcal{O}((\log n)^2)$. Moreover, there are other heuristic attacks that use Coppersmith's technique based on lattices to find d when $d < n^{0.292}$. More information on those can be found on [6, 7].

On the other hand, since, as we have seen, using a small secret exponent can be dangerous, one may be tempted to use a (very) large d to speed up the process. Although this sounds wrong when thinking about RSA being used in the positive representation (i.e., all values are always positive), if the exponents are in the symmetric representation, the computational cost of exponentiation can be substantially reduced by using small negative exponents (or equally very large positive exponents). More precisely, if d is small then the cost of computing $m^{-d} \bmod n$ is simply the computation cost of the exponentiation $m^d \bmod n$, followed by a $\bmod n$ inversion. In 2004, Hinek [8] proved that Wiener's continued fraction attack can be extended to very large private exponent RSA, with the following theorem:

Theorem. Let n be an RSA modulus with balanced primes[1] and let d be a private exponent satisfying $\sqrt{6}(\phi(n)-d) < n^{1/4}$. Given the public key (n, e), the private exponent can be recovered in time polynomial in $\log_2 n$.

Moreover, he proved that if the attacks from [6, 7] work for all $d < n^{\delta}$, then the attacks work also for $\delta > \phi(n) - n^{\delta}$.

Taking into consideration all of the above we will do an introduction of the Euclidean Attack on RSA. Generally, it too works by utilising the relation equation for e and d, along with Euclidean algorithm and some other computations to find the primes p and q. Although at first it looks complicated, in reality the thought process behind it is the same as in Weiner's attack: start from a relation of known and unknown variables and use a tool, in this case the Euclidean algorithm, along with the publicly known values e and n to find the solution. More precisely though, the conditions required for this attack are

(i) The public exponent e has the same order of magnitude as n.

(ii) One of the integers $k = \frac{(ed-1)}{\phi(n)}$ and $e - k$ has at most one-quarter as many bits as e.

About the first case of (ii), as k has at most one-quarter as many bits as e if and only if d has at most one-quarter as many bits as n, we see that this hypothesis is equivalent to that of Weiner's. Similarly, the second case of (ii) can be compared to Hinek's extension of Wiener's attack. Moreover, concerning its computation speed, the attack uses a deterministic algorithm for computing the factorization of an RSA modulus n in $\prime((\log n)^2)$ bit operations.

1.2 How it works

Let p and q be two primes, > 2 of the same size ($l(p) = l(q) = l$)) and set $n = pq$. Consider integers $e, d \in (1, \phi(n))$ such that $ed \equiv 1 \,(mod\, \phi(n))$. Then, as mentioned in the introduction, (n, e) is the public key and d is the private key for an RSA cryptosystem. Also, we set $a = n + 1 \bmod e$ and $\Delta = gcd(e, a)$.

It was mentioned before that the Euclidean algorithm will be used for this attack but never on which integers. Those integers will be e and a (we remind that $e¿a$). Continuing, the extended Euclidean algorithm will be used for e and a and its results will be denoted, as we will need them when stating the main theorem of Poulakis' paper:

[1] More on balanced primes can be found on [8], but for our interests it is merely a condition that doesn't impose on any of the results mentioned before.

We set $r_0 = e$ and $r_1 = a$, then there are pairs of integers (q_i, r_{i+1}) $(i = 1, \ldots, m)$ such that $r_m = \Delta$, $r_{m+1} = 0$ and

$$r_{i-1} = r_i q_i + r_{i-1},\ 0 < r_{i+1} < r_i.$$

Furthermore, we define integers s_i and t_i for $i = 1, \ldots, m+1$ as follows:

$$s_0 = 1, s_1 = 0 \text{ and } t_0 = 0, t_1 = 1$$

and for $i = 1, \ldots, m$,

$$s_{i+1} = s_{i-1} - s_i q_i \text{ and } t_{i+1} = t_{i-1} - t_i q_i.$$

It can be proven [9] that those integers t_i and s_i satisfy $|t_i| < e/r_{i-1}$, $|s_i| < a/r_{i-1}$ and

$$es_i + at_i = r_i\ (i = 0, \ldots, m+1).$$

Finally, we set $\mu_i = gcd(t_i, r_i)$, $t'_i = t_i/\mu_i$ and $r'_i = r_i/\mu_i$ $(i = 0, \ldots, m+1)$. The application of the Euclidean algorithm on the cryptanalysis of RSA is based on the following theorem:

Theorem 1. Let $e > n/c$, where c is an integer ≥ 2 and $k = (ed-1)/\phi(n)$. Let j be an index such that r_j is the largest remainder among them which are $< e^{3/4}$. The results below follow:

(i) Suppose that k is $\leq e^{1/4}/6\sqrt{c}$. Then, we have $\Delta < e^{3/4}$ and

$$k = |t'_j|, \quad p + q = (a + |t'_j|^{-1})\, mod\, e.$$

(ii) Suppose that $e - k$ is $\leq e^{1/4}/6\sqrt{c}$. Then, we have $\Delta < e^{3/4}$ and

$$k = |t'_j|, \quad p + q = (a + (e - |t'_j|)^{-1})\, mod\, e.$$

1.3 Why it works

In this section we will present the proof of Theorem 1 along with more analytic proof of the comparable efficacy between Poulakis's and Wiener's approaches.

1.3.1 Proof of Theorem 1

As mentioned before, we will start by the modulo congruence that connects e and d:

$$ed \equiv 1\, (mod\, \phi(n)).$$

From it we deduce that there exists an integer k such that $ed - k\phi(n) = 1$. Using this equality together with the fact that $\phi(n) = (p-1)(q-1) = pq - (p+q) + 1$ and $n = pq$ yields:

$$ed - k(n - (p+q) + 1) - 1 = 0,$$

from which we get:

$$k(n + 1 - (p+q)) + 1 \equiv 0\, (mod\, e).$$

Moreover, setting $y_0 = k$ and $x_0 = p + q$, we get:

$$1 + ay_0 - x_0 y_0 \equiv 0\, (mod\, e).$$

Suppose that $p < q$. Then $p^2 < p * q < n$ and thus $p < \sqrt{n}$. Further, since $l(p) = l(q) = l$, we have:

$$2^{l-1} + 1 \leq p < q \leq 2^{l-1} + \cdots + 1.$$

Thus, we get:

$$\begin{aligned} q - p &\leq 2^{l-1} + \cdots + 1 - (2^{l-1} + 1) \\ &\leq 2^{l-2} + \cdots + 2 \\ &< 2^{l-1} + 1 \\ &\leq p, \end{aligned}$$

and hence, $q < 2p$. Therefore, we obtain $x_0 = p + q < 3p < 3\sqrt{n} < 3\sqrt{ce}$, from $e > n/c$.

Case (i).
We assume that $y_0 \leq e^{1/4}/6\sqrt{c}$.
We suppose $\Delta \geq e^{3/4}$ and would like to prove the inverse by contradiction.
Firstly, since $\Delta = gcd(e, a)$ we get:

$$\begin{aligned} &1 - x_0 y_0 \equiv 0 \,(mod\, \Delta) \\ \Rightarrow &\exists \lambda \in \mathbb{Z} \text{ s.t.: } x_0 y_0 - 1 = \lambda \Delta \end{aligned}$$

Moreover, we have:

$$\begin{aligned} |x_0 y_0 - 1| &< 3\sqrt{ce} \cdot \frac{e^{1/4}}{6\sqrt{c}} - 1 \\ &< e^{3/4}/2 - 1 \\ &< e^{3/4} \end{aligned}$$

and thus

$$\begin{aligned} &|x_0 y_0 - 1| < e^{3/4} \leq \Delta \\ \Rightarrow &-\Delta < x_0 y_0 - 1 < \Delta \\ \Rightarrow &-\Delta < \lambda \Delta < \Delta \\ \Rightarrow &\lambda = 0. \end{aligned}$$

Hence, $x_0 y_0 = 1$, whence we get $x_0 = y_0 = 1$, which is a contradiction.
Therefore, $\Delta < e^{3/4}$.

Let now r_j be the bigger remainder $< e^{3/4}$, we have $r_{j-1} > e^{3/4}$ and thus, $|t_j| < e/r_{j-1} < e^{1/4}$. Furthermore, we have:

$$\begin{aligned} &t_j(1 + a y_0 - x_0 y_0) + s_j e y_0 \equiv 0 \,(mod\, e) \\ \Rightarrow &t_j + t_j a y_0 - t_j x_0 y_0 + s_j e y_0 \equiv 0 \,(mod\, e) \\ \Rightarrow &t_j + (t_j a + s_j e) y_0 - t_j x_0 y_0 \equiv 0 \,(mod\, e) \\ \Rightarrow &t_j + r_j y_0 - t_j x_0 y_0 \equiv 0 \,(mod\, e) \end{aligned}$$

We set $f(x, y) = t_j + r_j y - t_j x y$ and would like to prove $f(x_0, y_0) = 0$. At first, from the congruence above we have $e | f(x_0, y_0)$, and secondly we have $|f(x_0, y_0)| < e$ because:

$$\begin{aligned} |f(x_0, y_0)| &< e^{1/4} + e^{3/4} \cdot \frac{e^{1/4}}{6\sqrt{c}} + e^{1/4} \cdot \frac{e^{3/4}}{2} \\ &= e^{1/4} + \frac{e}{6\sqrt{c}} + \frac{e}{2} \\ &< e. \end{aligned}$$

Hence, $f(x_0, y_0) = 0$and thus, by dividing with $\mu_j = gcd(t_j, r_j)$, we obtain:

$$t'_j + r'_j y_0 - t'_j x_0 y_0 = 0$$

The above equality implies that $t'_j | r'_j y_0 \Rightarrow t'_j | y_0$, since $gcd(t'_j, r'_j) = 1$. Moreover, we also get $y_0 | t'_j$. Hence, we have $y_0 = |t'_j|$. Using this in the starting congruence $1 - ay_0 - x_0 y_0 \equiv 0 \, (mod\, e)$ yields

$$x_0 = (a + |t'_j|^{-1}) \, mod\, e$$

Case (ii).
We set $z_0 = -y_0 \, mod\, e$ and assume that $z_0 \leq e^{1/4}/6\sqrt{c}$.
We suppose $\Delta \geq e^{3/4}$ and would like to prove the inverse by contradiction.
The proof is similar to the one in case (i), only having $-z_0$ in the place of y_0. This way we get:

$$\begin{aligned} 1 + x_0 z_0 &\equiv 0 \, (mod\, \Delta) \\ \Rightarrow \exists \lambda \in \mathbb{Z} \text{ s.t.: } & x_0 z_0 + 1 = \lambda \Delta \end{aligned}$$

Moreover, we have $|x_0 z_0 + 1| < 1 + \frac{e^{3/4}}{2} < \Delta$ and thus we get $x_0 z_0 + 1 = 0$, which is a contradiction. Therefore, $\Delta < e^{3/4}$.

Let now r_j be the bigger remainder $< e^{3/4}$, we have $r_{j-1} > e^{3/4}$ and thus, $|t_j| < e/r_{j-1} < e^{1/4}$. Furthermore, we have:

$$\begin{aligned} t_j(1 - az_0 + x_0 z_0) - s_j e z_0 &\equiv 0 \, (mod\, e) \\ \Rightarrow t_j - t_j a z_0 + t_j x_0 z_0 - s_j e z_0 &\equiv 0 \, (mod\, e) \\ \Rightarrow t_j - (t_j a + s_j e) z_0 + t_j x_0 z_0 &\equiv 0 \, (mod\, e) \\ \Rightarrow t_j - r_j z_0 + t_j x_0 z_0 &\equiv 0 \, (mod\, e) \end{aligned}$$

We set $f(x, y) = t_j + r_j y - t_j xy$ and working similarly to before we finally get $z_0 = |t'_j|$. Using this in the starting congruence $1 - az_0 + x_0 z_0 \equiv 0 \, (mod\, e)$ yields

$$x_0 = (a + (e - |t'_j|)^{-1}) \, mod\, e,$$

which concludes the proof.

1.3.2 Equivalence with Wiener's attack

Finally, we shall give the proof of the "equivalence" between Poulakis' and Wiener's attacks:

Suppose that $n/(c-1) > e > n/c$ with $c \geq 2$. Then we have:

$$\frac{d}{k} = \frac{ed}{ek} = \frac{k\phi(n) + 1}{ek} < \frac{\phi(n)}{e} + \frac{1}{ek} < \frac{n-1}{e} + \frac{1}{ek} < \frac{n}{e} < c$$

Moreover, if we suppose $k \leq e^{1/4}/6\sqrt{c}$, then we get:

$$d < kc \leq \frac{\sqrt{c} e^{1/4}}{6} < \frac{\sqrt{c}}{6} \frac{n^{1/4}}{(c-1)^{3/4}}.$$

Hence, for $c = 10$, we get $d < n^{1/4}/3$.

On the other hand, supposing $d < n^{1/4}/3$ and using $\phi(n) < \sqrt{\frac{n}{2}} < \frac{n}{2}$ we have:

$$k = \frac{ed - 1}{\phi(n)} < \frac{ed}{\phi(n)} < \frac{2ed}{n} < \frac{2}{c-1} d < \frac{2e^{1/4}}{3(c-1)^{3/4}}.$$

Hence, for $c \geq 19$, we get $k \leq e^{1/4}/6\sqrt{c}$.

Therefore, we see that the approaches have comparable efficiencies. Moreover, we remind that Wiener's attack needs $\mathcal{O}((\log n)^3)$ bit operations while Poulakis' attack needs $\mathcal{O}((\log n)^2)$.

Chapter 2

Algorithm and Implementation

In this chapter, we will describe the EUCLID-ATTACK, which is a deterministic algorithm which uses the results of Theorem 1 in order to factorize n. Some remarks will be made concerning its steps and its computation speed will be presented (and proved). After this, two toy examples of use of the algorithm will be presented, one for each case of Theorem 1, whose intermediary values and output we got from our program.

2.1 EUCLID-ATTACK algorithm

As mentioned in the original paper, Theorem 1 and its proof yield the design of a deterministic algorithm that can efficiently compute the factorization of n. The algorithm, which was given the name "EUCLID-ATTACK", is the following:

* EUCLID-ATTACK algorithm *

· Input: An RSA public key (n, e) with $e > n \,/\, c$
· Output: The primes p and q (or FAIL)

1. Compute $a = (n+1) \, mod \, e$
2. Using the extended Euclidean algorithm for e and a, compute the biggest remainder r_j among them which are $< e^{3/4}$, as well as the associated integers s_j, t_j such that $s_j e + a t_j = r_j$.
3. Compute $\mu_j = gcd(t_j, r_j)$ and next $t'_j = t_j \,/\, \mu_j$.
4. Compute $\beta_1 = (a + |t'_j|^{-1}) \, mod \, e$ and next the solutions u_1 and v_1 of equation $x^2 - \beta_1 x + n = 0$. If u_1 and v_1 are positive integers, then output (u_1, v_1). Otherwise, go to the next step.
5. Compute $\beta_2 = (a + (e - |t'_j|)^{-1}) \, mod \, e$ and next the solutions u_2 and v_2 of equation $x^2 - \beta_2 x + n = 0$. If u_2 and v_2 are positive integers, then output (u_2, v_2). Otherwise, output FAIL.

Theorem 2. Let $e > n/c$, where c is an integer ≥ 2 and $k = (ed-1)/\phi(n)$. Suppose that k or $e - k$ is $\leq e^{1/4}/6\sqrt{c}$. Then, the above algorithm computes correctly the primes p and q in time $\mathcal{O}((\log e)^2)$ bit operations.

Proof. In order to compute the time complexity of the algorithm we account the following:

- The extended Euclidean algorithm used on the 2nd step needs $\mathcal{O}((\log e)^2)$ bit operations.
- The computation of the $gcd(t_j, r_j) = \mu_j$ and t'_j on the 3rd step also needs the use of the Euclidean algorithm with time $\mathcal{O}((\log e)^2)$ bit operations.
- The computation of the inverses for β_1 and β_2 also requires $\mathcal{O}((\log e)^2)$ bit operations.

- The computation of the solution of the two quadratic equations also takes $\mathcal{O}((\log e)^2)$ bit operations.

Consequently, the time complexity of the algorithm is $\mathcal{O}((\log e)^2)$ bit operations. □

2.2 Experimental Results

In this section we will present two examples that correspond to each case of Theorem 1, in order to showcase the algorithm. The results were found using the accompanying executable program for the chosen n and e in each case. Moreover, in order for the solution to fit properly inside the GUI (Graphical User Interface) window, the modulus n was chosen to be a 512-bits number. The algorithm in the .nb file can give a result for much bigger n but the executable would have to be made again in order to show the results problem.

For both examples we will choose the same 256-bits primes p and q.
Let the two primes be:

$$p = 89231715549040888301567357716904006861451079875545083716711784621251557817749$$
$$q = 94841300244835204089187442585337074762553173477433230632506238441913090478509$$

From them, we get the modulus $n = pq$:

$$n = 84628519257483168873531866424128792688524492271679568476565354549532800090$$
$$54764041966937146611183036071299428938033981872593180570454814757911422325624 1$$

Furthermore, we calculate $\phi(n)$:

$$\phi(n) = 84628519257483168873531866424128792688524492271679568476565354549532800090$$
$$08363567403875495373721075560410753207756335565372953491355330124515949574959984$$

2.2.1 Example with a small private exponent

We first remind that the algorithm works for $d < n^{1/4}/3$, so we have deliberately chosen d to be really close to this limit. More precisely, d is a 127-bit number, with

$$d = 10110157722388977465787129390998 4870399$$

Thus, $e = d^{-1} \, mod \, \phi(n)$ will be a 509-bit number, with

$$e = 11524363617821508909071871668005015761672643282586144665861195480767486297 51$$
$$30113587137419044714967867055374176656246918460606355969688937764033985102134 3$$

Therefore we will now use the algorithm to compute the factorization of n, where (n, e) and d are, respectively, the public and private keys for an RSA scheme.

We will start by calculating $a = (n + 1) \, mod \, e$, which will be

$$a = 39579739327326065100287647480936823568159892935765558155369861841603960074 9$$
$$43968932005003833606407966683680192344305552648348688782632250409673526610684 1$$

We apply the Euclidean Algorithm for $r_0 = e$ and $r_1 = a$, and calculate the remainders. Of them, the biggest remainder $< e^{3/4}$ is r_j for $j = 77$, with

$$r_j = 25342430791866723821720418728283847802523253822522027918099125958286 48$$
$$82658371792678414440459164660838556621192957 1$$

The corresponding s_j and t_j are

$$s_j = -472839901632716941393238625199193296 1$$
$$t_j = 13767596886794657891991272522570419634$$

Moreover, we have $\mu_j = gcd(r_j, t_j) = 1$ and thus, $t'_j = t_j$.
Thus we will now compute $\beta_1 = (a + |t'_j|^{-1}) \, mod \, e$ along with the solutions u_1 and v_1 of equation $x^2 - \beta_1 x + n = 0$:

$$\beta_1 = 184073015793876092390754800302241081624004253352978314349218023063164648296258$$
$$u_1 = 892317155490408883015673577169040068614510798755450837167117846212515578177 49$$
$$v_1 = 948413002448352040891874425853370747625531734774332306325062384419130904785 09$$

For completeness, we also compute $\beta_2 = (a + (e - |t'_j|)^{-1}) \, mod \, e$ along with the solutions u_2 and v_2 of equation $x^2 - \beta_2 x + n = 0$:

$$\beta_2 = 791594786546521302005752949618736471363197858715311163107397236832079201498 6$$
$$95305624306200579737404533371362765262106799613995461303426985130305883917424$$
$$u_2 \simeq 10.69$$
$$v_2 \simeq 7.91e + 15$$

Therefore, we see that the solutions of the first equation are integers and are thus the primes p and q. Note also that $8e > n$ and so, $c = 8$. Furthermore, $k < e^{1/4}/6\sqrt{8}$, where

$$k = 13767596886794659055372183688120893440$$

2.2.2 Example with a (very) large private exponent

This time we just chose a d really close to n in order to showcase how the algorithm works for (very) large exponents. More precisely, d is now a 512-bit number, with

$$d = 846285192574831688735318664241287926885244922716795684765653545495328000900 8$$
$$3635674038754953737210755604107532077563355653729534913553301245159495749599 79$$

Thus, $e = d^{-1} \, mod \, \phi(n)$ will be a 512-bit number, with

$$e = 677028154059865350988254931393030341508195938173436547812522836396262400720 6$$
$$690853923100396298976860448328602566205068452298362793084264099612759659967987$$

Therefore we will now use the algorithm to compute the factorization of n, where (n, e) and d are, respectively, the public and private keys for an RSA scheme.

We will start by calculating $a = (n + 1) \, mod \, e$, which will be

$$a = 169257038514966337747063732848257585377048984543359136953130709099065600180 1$$
$$8567864965689751671349699123843917231752713664275690126202840479663545632882 55$$

We apply the Euclidean Algorithm for $r_0 = e$ and $r_1 = a$, and calculate the remainders. Of them, the biggest remainder $< e^{3/4}$ is r_j for $j = 3$, with

$$r_j = 736292063175504369563019201208964326496017013411913257396872092252658593185033$$

The corresponding s_j and t_j are

$$s_j = -1$$
$$t_j = 4$$

Moreover, we have $\mu_j = gcd(r_j, t_j) = 1$ and thus, $t'_j = t_j$.
Thus we will now compute $\beta_1 = (a + |t'_j|^{-1}) \, mod \, e$ along with the solutions u_1 and v_1 of equation $x^2 - \beta_1 x + n = 0$:

$$\beta_1 = 338514077029932675494127465696515170754097969086718273906261418198131200360352949997734407424187918502446654236472653847950215971089135007286954447828025 2$$

$$u_1 \simeq 2.5$$

$$v_1 \simeq 3.3e + 15$$

For completeness, we also compute $\beta_2 = (a + (e - |t'_j|)^{-1}) \, mod \, e$ along with the solutions u_2 and v_2 of equation $x^2 - \beta_2 x + n = 0$:

$$\beta_2 = 184073015793876092390754800302241081624004253352978314349218023063164648296258$$

$$u_2 = 892317155490408883015673577169040068614510798755450837167117846212515578177 49$$

$$v_2 = 948413002448352040891874425853370747625531734774332306325062384419130904785 09$$

Therefore, we see that the solutions of the second equation are integers and are thus the primes p and q. Note also that $2e > n$ and so, $c = 2$. Furthermore, $e - k < e^{1/4}/6\sqrt{2}$, where

$$k = 677028154059865419483963306652404392980716041637586912848286083210696473141407636805045144005184669507883892067280450792995903557604926857703641292210176 0$$

References

[1] Dimitrios Poulakis. *An Attack on Small Private Keys of RSA Based on Euclidean Algorithm*. Cryptology ePrint Archive, Paper 2019/283. `https://eprint.iacr.org/2019/283`. 2019. URL: `https://eprint.iacr.org/2019/283`.

[2] M.J. Wiener. "Cryptanalysis of short RSA secret exponents". In: *IEEE Transactions on Information Theory* 36.3 (1990), pp. 553–558. DOI: `10.1109/18.54902`.

[3] Andrej Dujella. "Continued fractions and RSA with small secret exponent". In: *CoRR* cs.CR/0402052 (2004). URL: `http://arxiv.org/abs/cs/0402052`.

[4] Eric R. Verheul and Henk C. A. van Tilborg. "Cryptanalysis of 'Less Short' RSA Secret Exponents". In: *Applicable Algebra in Engineering, Communication and Computing* 8 (1997), pp. 425–435.

[5] Alexander May. "Using LLL-Reduction for Solving RSA and Factorization Problems". In: *The LLL Algorithm*. 2010.

[6] Johannes Blömer and Alexander May. "Low Secret Exponent RSA Revisited". In: *Revised Papers from the International Conference on Cryptography and Lattices*. CaLC '01. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 4–19. ISBN: 3540424881.

[7] Dan Boneh and Glenn Durfee. "Cryptanalysis of RSA with private key d less than N0.292". In: *Information Theory, IEEE Transactions on* 46 (Aug. 2000), pp. 1339–1349. DOI: `10.1109/18.850673`.

[8] M. Jason Hinek. *(Very) Large RSA Private Exponent Vulnerabilities*. CACR Technical Report 2004-01. Centre for Applied Cryptographic Research, University of Waterloo, 2004.

[9] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge: Cambridge University Press, 2005. DOI: `10.1017/CBO9781139165464`.