CS 13: Mathematics for Computer Scientists — Spring 2023

Applications Homework 00 (due Friday, April 28)

Directions: *Write up carefully argued solutions to the following problems. Your solution should be clear enough that it should explain to someone who does not already understand the answer why it works. You may use results from lecture and previous homeworks without proof.*

In this applications assignment, you will explore various ways that RSA can completely break if the keys aren't chosen carefully!

0. Message Size (20 points)

The first attack will be based on the *size of the message* being small. You may note that this attack is also dependent on e being small, but, in most RSA implementations e is usually chosen to be small on purpose.

Your job is to decrypt `encrypted_message` given that it was encrypted using **K0**. To do this, you will want to take the eth root of the encrypted message. Consider using the two argument `pow` function in Python and rounding the output with `round`.

K0:

```
N = 27757098857064826737036098474031627955533055737419281107143225982125460644011305883560832121497315397135189973176627544089920825316535758662004057452835043977047864453944520289174206459357701613961369669485256463748894829252863628579408193091008795364134277299525034759699161762164830167608665045842374063326114605306634033831903009439249232283316963697972003458183737076206646022007178761689286628727519302964840911264216341571258173723529491426041100641364707054325838941782146393635016353550607318640978548384452185904400903188079534247131338435241316750644507074805726851366426169406208730117954624324667689676753

e = 3

encrypted_message = 109599938775622399587345269212768768
```

Note that to convert `num` (a number) to `m` (a string representing the message), you can use the following line in Python: `m = bytes([x for x in num.to_bytes(1000, byteorder='little') if x != 0])`.

(a) [10 Points] Write and submit Python code that recovers the original message in English **and** also submit the actual decrypted message.

(b) [10 Points] Write and submit an explanation of why the message has to be short for this attack to work.

1. Wiener's Attack (80 points)

The second attack will be significantly more sophisticated than the previous one (though, surprisingly not that much more code). Consider the following (weird) theorem:

Wiener's Theorem

Given an RSA key (N, e, d, p, q) with the following properties:

- $q < p < 2q$
- $d < \frac{N^{\frac{1}{4}}}{3}$

An attacker can efficiently recover the entire private key: (d, p, q) from just the public part: (N, e).

While this theorem looks scary and arbitrary, you'll investigate it in two ways in this assignment. First, you'll finish a proof of the theorem. Then, you'll implement the attack on some insecure keys.
Before we can get into either the implementation or proof of the theorem, we need make a quick digression to the land of continued fractions! Continued fractions are a way of representing a real number as a sequence of integers. Given a number $x \in \mathbb{R}$, it's continued fraction representation is of the form:

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

where $a_0, a_1, a_2, a_3, \ldots$ are known as the *coefficients* of the continued fraction.
The *convergents* of a continued fraction are the best rational approximation of the number x which can be obtained by truncating the continued fraction. The n-th convergent is of the form:

$$\frac{h_n}{k_n} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \frac{1}{a_n}}}},$$

Now we are ready to start proving Wiener's theorem! First, note that $ed \equiv_{\phi(N)} 1$. So, there is a $k \in \mathbb{Z}$ such that $ed - k\phi(N) = 1$. Dividing both sides by $d\phi(N)$, we get:

$$\frac{e}{\phi(N)} - \frac{k}{d} = \frac{1}{d\phi(N)}$$

A Theorem By Legendre

If $\left|a - \frac{b}{c}\right| < \frac{1}{2c^2}$ and $\gcd(b, c) = 1$, then b/c appears as some convergent of the continued fraction of a.

(a) [10 Points] Use the above to prove that Legendre's theorem applies for $a = \dfrac{e}{\phi(N)}$ with $\dfrac{b}{c} = \dfrac{k}{d}$. That is, show

$$\gcd(k, d) = 1 \qquad \text{and} \qquad \left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)} < \frac{1}{2d^2}.$$

You'll need to use the givens, from above: $d < \frac{N^{\frac{1}{4}}}{3}$ and $q < p < 2q$.

Unfortunately, for us, $\phi(N)$ is still something we cannot compute. Instead, we'll try switching the denominator on the left to just plain N by showing the resulting difference is still within the threshold.

That is, we want to show:

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2}$$

But first, a lemma!

Lemma 1. $|N - \phi(N)| < 3\sqrt{N}$.

(b) [10 Points] Prove Lemma 1 using the given that $q < p < 2q$.

Now, use the first lemma to prove another lemma!

Lemma 2. $\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{3k}{d\sqrt{N}}$.

(c) [10 Points] Prove Lemma 2 using Lemma 1 and the proof of Wiener's Theorem on Wikipedia. We believe the proof on there is so convoluted that it is a good usage of your time to try to understand it instead of duplicate it. Note that you must justify every step in this part (which Wikipedia does not do). You may not reference the proof on Wikipedia directly though.

And again, another lemma!

Lemma 3. $k < d$.

(d) [10 Points] Prove Lemma 3.

(e) [10 Points] Complete the proof of the new version of the claim by using Lemma 2 and Lemma 3. That is, show:

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2}$$

Finally! Now that we've satisfied the constraints of Legendre's theorem above, we can apply it to find $\frac{k}{d}$ as a convergent of $\frac{e}{n}$.

Now, we can get to coding! You can now write a Python program to implement Wiener's Attack on some unsuspecting keys. This attack will have three pieces:

I. Find the coefficients of the continued fraction of e/N.

1. Initialize $r = e/N$. Sequentially take $i = \lfloor r \rfloor$, add i to your list of coefficients, and set $f = r - i$. If $f = 0$, stop. Otherwise, set $r = 1/f$ and repeat.
2. Use the `fractions` library in Python to wrap your r in a `Fraction` object - this will avoid compounding floating point errors and make your life much easier.

II. Find the convergents of the continued fraction of e/N.

1. Given a list of coefficients a_n for $n \geq 0$ (which you calculated in step I.), the convergents $\frac{h_n}{k_n}$ can be found according to the recurrences:

$$\begin{aligned} h_{-2} &= k_{-1} = 0 \\ k_{-2} &= h_{-1} = 1 \\ h_n &= a_n h_{n-1} + h_{n-2} \\ k_n &= a_n k_{n-1} + k_{n-2} \end{aligned}$$

III. For each convergent $c = a/b$:

1. Calculate $\texttt{potential\_phi} = (eb - 1)/a$. Use integer division (which is // in Python).
2. Solve this quadratic equation using the Python code below.

$$x^2 - (N - \texttt{potential\_phi} + 1)x + N = 0$$

```
def solve(a, b, c):
    from decimal import Decimal, getcontext
    getcontext().prec = 1000
    a = Decimal(a)
    b = Decimal(b)
    c = Decimal(c)
    return (int((-b + (b**2 - 4*a*c).sqrt())/(2*a)), int((-b - (b**2 - 4*a*c).sqrt())/(2*a)
        ))
```

3. Check if the two solutions to the quadratic equation multiply together to make N. If so, we found a factorization!

(f) [30 Points] For each of the three RSA keys below, recover p and q. You should submit the actual values as well as your Python code.

K1:

```
N = 90581

e = 17993
```

K2:

```
N = 13275993855447827057905575316152928933803559713561038486953556955295528774181862206805700546600906242564213981921365567995067466223388392052345549916385464010898896186245856602835178578423440480066592376460007112733599649807691968226433519496264529549808697687991163817510668677624950462967083150968059963277221081933239285021940877379067473095833217383645672515635215388294566119680851302961902254719359648080957083556347400462816497046344173078168361688270854607093902592058539570737961357033856659209349287236681426501482268591633258592399958227708167643570575052735251331518114924442746592771217694695504776276961
```

```
e = 68694546789862599432956723985321936702331966297416426541484821337615591963906525585042771062213382688701299757720554618956684218561467630031055168651041847552009727022650039437212657443740973443542414513535135018578785001450631481533280583956881857196947606225136400952974448956031279830234286882069839601954548091097738205903547685623413058371084466067570963727225659906303503525805613242980774778535461448887583631727956698766493838895589791572171884567562561908917134369632516464408070316126918264253106051479057988714187015111852070580754712993481203973484260301529950541309705664160484410287300663691765760001 7
```

(continued on next page)

K3:

N = 881204353778399283459237540223487039664182534173570129964717625640659973296805177089688216092300383360729996480188501512542507236011360648139710988057569803880577166678963426806064048958935291877637033951269757411504670794516676893555215201842485401842150724201669018969026691497693717069478120762829514460032193796142664544017181243353248683482418479004213375582443275235822165900630749458501399763956224105847817023488413871651570063650248401641627313797571604036378045757770851191610562253689697787048103389468234953507029989406441911597535510052711371915587581168102080247666793164559467109431802509994964838334 7

e = 322830134230326642116911731576026575843842586065733302307582959755239521201779643691188427528796639040761780293872417836694219446620572847037204321660893945683995480478684517557221310812943679586038984510025856653068430096997049932865556403903266282929939235605711305132183134166925740265104924412319548787185625890982430597689658496301600299407161923797664999697852829012478549822125235360847482104396890849982225289128027040565643775186536046035695082005342771616508485300964082999055800362670762098687387374313665518028572378980143232298939260678427805985832954745888548225228129691496673973128433757400155631336 1