CS641A (Modern Cryptology)
Chapter-6 Report

Instructor: Prof. Manindra Agrawal

Group Name: CMEN

Names:	Pratyush	170500
	Pravar Deep Singh	160508
	Tushar Garg	160749
	Umesh Meena	160755

Date of Submission of Report: 15/05/2020

We found the assignment in the mail, which stated as follows:

n = 8436444373572503486440255453382627917470389343976334334
3863260342756678609216895093779263028809246505955647572176682
2669445270008816481771701417554768871285020442403001649254405
0583034399062292019095993486695656975343316520195164095148002
6588738853928338105393743349699444214641968202764907970498260
0857517093

This door has RSA encryption with exponent 5 and the password is

5885119081935571454727589955844171566374613984724607561922
7074533865700705569837874063774277536176889970088885808705
06626143183054430644488980265035567576103429384907413616436
962850518672602785678969919273519645573749776196447636332298
96668511752432222528159214013173319855645351619393871433455550
581741643299

From reading the assignment we thought we have to use the fact that the encryption is done with exponent 5. In the class, the method taught for small exponent was based on the fact that we know some prefix of the message encrypted using small exponent RSA but by no means, we were aware of any such information so far. So, we start with simple factorization techniques that might work and try them.

Method to Factorize n:-

1. **Fermat Factoring Algorithm:-** This method is based on the fact that an odd number can be written as a product of two numbers. In particular, n = pq can be written as

$$n = pq = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$$

Let k be the smallest positive integer so that $k^2 > n$, and consider k^2 - n. If this is a square, we can factor n; if $k^2 - n = h^2$, then n = (k - h)(k + h). If it is not a square, increase the term on the left by one and consider $(k + 1)^2 - n = h^2$. If this is a square, n factors. If not, then we keep incrementing k and eventually, we will find 'h' so that k^2 - n factors. If one does not find 'h' s.t. k^2 - n is not a perfect square, then the number 'n' is prime. Fermat's Factorization Algorithm works well if the factors are roughly the same size.

We ran this method for about 4 hours, but we were not able to factorize n. So, it might have been the case that the factors of 'n' were not close enough to be found by this algorithm.

2. **Pollard p - 1 Factorization Algorithms:-** As taught in the class, this method is based on the assumption that p-1 is a smooth number i.e. it has a small prime divisor, where p is one of the factors of n.

Let r be the number which is divisible by p-1. Assume r = (p - 1)j and we also assume $1 < a < p - 1$. Now consider

$$a^r = a^{(p-1)j} = (a^{p-1})^j = 1 \pmod p$$

Hence, a^r - 1 is divisible by p.

$$\gcd(a^r - 1, n) = p \text{ or } n.$$

If we obtain the gcd to be n, then we choose another 'a' and try again.

First, we ran this method for r = 1000! but the program terminated with no result, meaning our guess for r was too small. Then again, we ran this method for 4 hours with r = 1000000, but we obtained no result and the program kept running for hours and in the end, we terminated the program by manual intervention.

3. **Pollard Rho Method:-** In this method, we tried to find one of the factors of n by using the following algorithms.

   We take the function $f(x) = x^2 + 2$, and do as following:

```
$x_1 = 2, x' = f(x)$
while(1):
    $d = gcd(x - x', n)$
    If $d < n$:
        Break
    If $d == 1$:
        $x = f(x)$
        $x' = f(f(x'))$
```

This algorithm is supposed to work in $\sqrt{p}$ steps, but since the number is very large we have to count for the modulo operation as well and multiplication of big numbers, all of which might be the reason because of which the program was not able to give the factorization of n while running for hours.

4. **Mersenne Prime Attack:-** Prime numbers of the form $2^p - 1$ are known as Mersenne prime numbers. This attack is based on the fact that one of p and q is the Mersenne prime number. Only 51 such prime numbers have been identified to date in which the smallest is 3 and the largest is $2^{82,589,933} - 1$. So we check all these 51 possibilities if any of those divides n completely. Unfortunately, none of them could divide n, hence the attack failed.

5. **Wiener Attack:** Wiener Attack is applied mainly when the prime factors of n are close enough and the value of d is small enough as said in Wiener's theorem which states that:

"Let n = pq with q < p < 2q. Let $d < \frac{1}{4}n^{1/4}$. Given (n, e) with ed $\equiv 1$ mod $\varphi(n)$. Then, the attacker can efficiently recover d."

In our case, we have e = 5, and n is very large so it's easy to observe that Wiener's theorem is inapplicable. However, the condition in Wiener's theorem is just a sufficient condition and not a necessary one. So we anyway tried it.

$$ed = 1 + k \cdot \varphi(n) \qquad (1)$$

Since n is very large, we can approximate $\varphi(n)$ as n, and hence, we can write $1 + k \cdot \varphi(n)$ as $k \cdot n$.

Hence, we get,

$$\frac{k}{d} \approx \frac{e}{n}.$$

Also, from (1), we see that k and d are coprime. So, to find these, we will look for the continued fractions of $\frac{e}{n}$, and then we will try each value of the pair (k, d) so obtained to calculate the value of $\varphi(n)$ from equation (1). If this value comes out to be non-integral, we will discard these values of k and d and look for the next continued fraction. If it is integral, then we will find the roots of the following quadratic equation

$$x^2 - (n - \varphi(n) + 1)x + n = 0.$$

If these roots are integral, then they will be the factors of n. If not, we will discard these values of k and d, and look for the next continued fraction.

Justification for the fact that this equation has p and q as roots:

The equation $x^2 - (p + q)x + pq = 0$, clearly has roots p and q. But $\varphi(n) = (p - 1)(q - 1) = pq - p - q + 1 = n - p - q + 1$. Hence, the result follows.
We ran the code and it stopped as soon as it started. This method failed to produce desired results.

After failing from all the 5 methods mentioned above, we started looking for some other approach or hints on the internet. Then we stumbled upon a blog, which was related to this field. After reading the blog, which said, "*For example, the message is always something like "the password today is: [password]".*", we finally realized that we might have been given the partial password information in the mail itself. So, by applying the same analogy as the blog, we thought the entire line in the mail might have been the partial password exposure to us. So, we went for Coppersmith Attack then.

Coppersmith Attack:-

According to Coppersmith's theorem,

"Let N be an integer and $f \in Z[x]$ be a monic polynomial of degree d over the integers. Set $X = N^{1/d-e}$ for $\frac{1}{d} > \varepsilon > 0$. Then, given $\langle N, f \rangle$ attacker, Eve, can efficiently find all integers $x_o < X$ satisfying $f(x_o) \equiv 0 \pmod{N}$. The running time is dominated by the time it takes to run the LLL algorithm on a lattice of dimension O(w) with $w = min\{\frac{1}{\varepsilon}, log_2 N\}$."

This theorem states that the existence of an algorithm that can efficiently find all roots of f modulo N that are smaller than $X = N^{1/d}$. As X gets smaller, the algorithm's runtime will decrease. This theorem's strength is the ability to find all small roots of the polynomials modulo a composite N.

Now, first of all, we assumed that we have the prefix of length long enough such that the unknown suffix is comparatively small to be found by the Coppersmith method. Now, from the mail we assumed that the prefix of the message is "This door has RSA encryption with exponent 5 and the password is", and the message is converted to a number by changing each character to corresponding 8-bit number, as it was the genuine choice since we had to represent small as well as capital letters of the message into a number. Now, we went through the Coppersmith method to find the unknown suffix of the message. Let the prefix of the message, after converting it to corresponding binary form, is P and assume that we don't know the last 'n' bits of the message then:

$$M = P \cdot 2^n + x$$

So, the polynomial equation that we get is

$$M^e = C \pmod N$$
$$(P \cdot 2^n + x)^e = C \pmod N$$

So, We have got a polynomial of degree e in terms of an unknown x, where e, C, N, and P are known. The solution to this problem is exactly similar to what we did in the class, but with e = 5 instead of e = 3. So, using the LLL lattice method, we want to find a root x for the above polynomial s.t. $x < 2^n$.

We can find the vector of shortest magnitude up to a constant factor and using that vector we can find the root of the equation modulo N and that will be the solution if our $x < N^{1/e}$ as stated in the theorem above.

Now we had one problem:- The value of n is not known beforehand, so we had to try all the values in the range $[1, N^{1/e}]$. Now to find the vector of the shortest magnitude, and solution of the polynomial we took help from Sage, an open-source mathematics software.

Finally, after very few numbers of iterations, we obtained the message to be "**This door has RSA encryption with exponent 5 and the password is tkigrdrei**". Hence, our algorithm found the suffix of the message to be " tkigrdrei". So, we found the password for level-6 to be '**tkigrdrei**'.

Explanation of the Codes:-

1. The file ‘fermat.py’ contains the code of the Fermat factorization method.
2. The file ‘pollard.py’ contains the code of the pollard p-1 method.
3. The file ‘pollard_rho.py’ contains the code of the Pollard rho method.
4. The file ‘Wiener_Attack.py’ contains the implementation of Wiener’s attack using the continued fraction.
5. The file ‘coppersmith.sage’ contains the implementation of Coppersmith Attack and to run this file we used the site mentioned in reference 6 (use ‘sage’ language on the site).

Reference:-

1. http://www.hri.res.in/~kalyan/lecture2.pdf
2. https://www.nku.edu/~christensen/Mathematical%20attack%20on%20RSA.pdf
3. https://en.wikipedia.org/wiki/Wiener%27s_attack
4. https://www.cryptologie.net/article/222/implementation-of-coppersmith-attack-rsa-attack-using-lattice-reductions/
5. https://en.wikipedia.org/wiki/Coppersmith%27s_attack
6. https://sagecell.sagemath.org/