ASSIGNMENT 6

CS-641

TEAM- Bullshot

Members –

Aman Tayal (180074)

Rohit Ranjan (180629)

Rishabh Kothary (180608)

For this assignment we needed to decipher a low exponent RSA cipher where e = 5. We were given the following password =

58851190819355714547275899558441715663746139847246075619270745338657007055698378740637742775361768899700888858087050662614318305443064448898026503556757610342938490741361643696285051867260278567896991927351964557374977619644763633229896668511752432222528159214013173319855645351619393871433455550581741643299

The value of N given was:

84364443735725034864402554533826279174703893439763343343863260342756678609216895093779263028809246505955647572176682669445270008816481771701417554768871285020442403001649254405058303439906229201909599348669565697534331652019516409514800265887388539283381053937433496994442146419682027649079704982600857517093

To solve it we referenced the paper “Finding Small Roots of Univariate Modular Equations” by Nicholas Howgrave (attached in the folder).

We were given some hint about the padding scheme and we tried various padding schemes like:

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

“This door has RSA encryption with exponent 5 and the password is\n”

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

For each padding scheme, first we converted them to their binary equivalent (using the ascii-256 mapping) and then to their decimal system equivalent and we call it ‘p’.

We denote the original password as ‘m’. And then we tried various padding schemes of the form (2lp + m) where l varied from (1 to 300) as n was a 308-digit number and such a padding scheme was discussed in class. Let us denote the ciphered password by C. So, we basically needed to solve for m in the equation:

P(m) = (2lp + m)e = C (mod N).

Here a,n,e are known to us and we have a pretty good idea about p. And we can guess the range of l possible is from around 1 to 308. As the possibilities of l and p are not large, we can brute force it down. Now in order to solve the modular polynomial the basic strategy was to find a smaller polynomial with m as a root and the we can solve it in Z rather than Zn. As n was around 1024-bit number factoring it is not feasible so our approach was to construct a lattice of polynomials and find a smaller root using LLL algorithm. We see that our polynomial is e =5 degree polynomial which refer to as ‘k’ to be consistent with the paper we referenced.

Then we choose a value of h>1 and a value of X with an assumption that m < X. We then construct a lattice M where the value of Mij is equal to eijXj-1, where eij is the coefficient of xj-1 in the polynomial: qu,v(x) = Nh-1xuP(x)v

where v = floor((i-1)/k) and u = i-1 – kv. An important thing to notice is that qu,v(m) = 0 (mod Nh-1).

After applying LLL lattice reduction algorithm to the lattice M we are able to find a smaller polynomial which can be solved over Z with the condition m < X. The analysis is given in the paper. The larger the value, the larger X we can find and hence larger roots can be captured. However, M is a (hk, hk) dimension lattice thus it becomes computationally difficult as we increase the value of h. We choose a value of h=3 for finding m which gave us a bound-on X ~ N1/9 (Details given in the paper).

Then after trying several values of p and l, we saw that we got the correct answer for

p = “This door has RSA encryption with exponent 5 and the password is ”

and l = 72.

We found the root of the small polynomial to be 2147562143725930046825.

Which when converted to binary becomes:

11101000110101101101001011001110111001001100100011100100110010101101001

Which when converted to ASCII text becomes “tkigrdrei” which was the correct password.

The code is attached in the folder.