CSE 534 HOMEWORK 1

PART A:

I have tried to implement a tool in python like LINUX dig command. The program takes two parameters i.e. domain name and type of record. The tool works for A, MX, and NS records. The tool also resolves the CNAME and displays the output as expected that of dig tool. The libraries used for developing the tool are dnspython. The major drawback time of the tool is that it does not provide the caching mechanism that LINUX dig tool provides. This results in longer query times than that of LINUX dig tool.

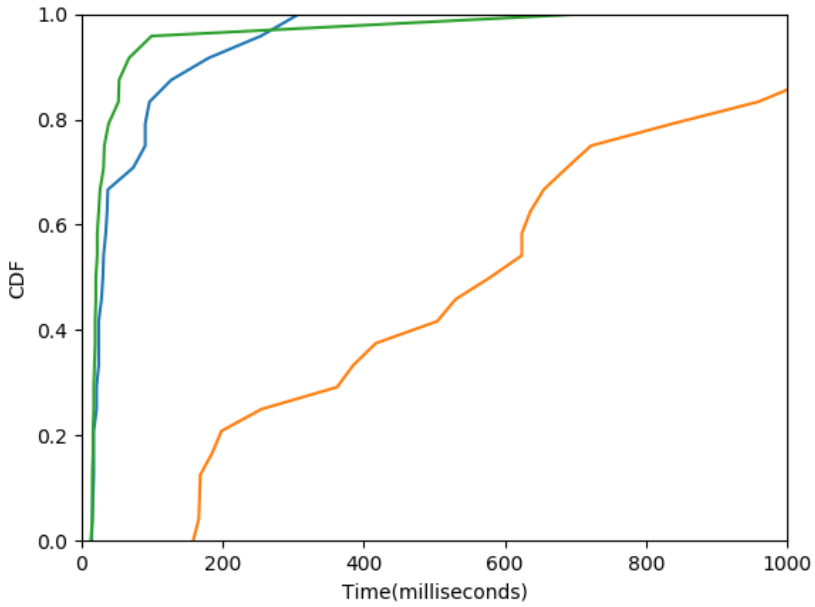
The tool is developed to provide support for corner cases such as google.co.jp. When we resolve googl.co.jp once we get the CNAME and thus we need to resolve the mapping of CNAME to ip address and hence we need an extra pass to resolve google.co.jp.

PART B:

The part B is implementation of dnssec with existing part A implementation. The code for Dns Sec is appended with the code in part A. In dnssec implementation, we get the child and parent pointers for a given domain. We query them using dns.query.tcp (two-way verification) and use the responses generated to create the DS, RRSIG (DNS HASHED), NS, RRSIG, DNSKEY. Then we perform the two-step validation to validate 1. DNSKEY and RRSIG (DNS HASHED) and 2. DNS and RRSIG. After the successful two validations we then validate the Ds record of child and parent by creating a hashed ds record for child and matching it with the parent. This is called the chain of trust. If this works well, then we further resolve the query as per the mechanism implemented in part A.

PART C:

The tool developed in part A was checked by querying top 25 websites from Alexa.com for 10 times. The whole process is being automated by coding the test part. The average resolution time was calculated using google public dns resolver, local dns resolver and mydig.py (tool developed). Then the CDF was calculated for the data and plotted against time in the graph below. The behavior of google resolver (in green) and local resolver (in blue) is quite similar. The mydig tool (in red) takes comparatively longer time to execute queries due to lack of caching mechanism which is evident form graph.



The tool displays the average time for different sites as:

The average time for: Google.com is 0.170199990273 msec

The average time for: Youtube.com is 0.17619998455 msec

The average time for: Facebook.com is 0.177300000191 msec

The average time for: Baidu.com is 0.821800017357 msec

The average time for: Wikipedia.org is 0.851499986649 msec

The average time for: Reddit.com is 0.164300012589 msec

The average time for: Yahoo.com is 0.419499993324 msec

The average time for: Google.co.in is 0.559800004959 msec

The average time for: Qq.com is 1.5256000042 msec

The average time for: Taobao.com is 0.243499994278 msec

The average time for: Amazon.com is 0.424800014496 msec

The average time for: Tmall.com is 0.236599993706 msec

The average time for: Twitter.com is 0.167899990082 msec

The average time for: Google.co.jp is 0.562800002098 msec

The average time for: Instagram.com is 0.202500009537 msec

The average time for: Live.com is 0.23599998951 msec

The average time for: Vk.com is 0.87539999485 msec

The average time for: Sohu.com is 0.898600006104 msec

The average time for: Sina.com.cn is 1.34110000134 msec

The average time for: Jd.com is 0.62460000515 msec

The average time for: Weibo.com is 0.476999998093 msec

The average time for: 360.cn is 2.2251999855 msec

The average time for: Google.de is 0.562400007248 msec

The average time for: Google.co.uk is 0.639500021935 msec

The average time for: Google.com.br is 0.566400003433 msec